

网页开发手记——486 个 JavaScript 网页特效详解

吴绍兴 韩峰 胡可 编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书涵盖了目前网络开发涉及的所有方向，从页面、文本、窗口、鼠标、日期时间等基本应用，到图像、滚动条、进度条、网络验证、文件处理等深入应用，包括目前最流行的异步传输、Property 框架、Ajax 和 DOM 等高级技术。全书共分 23 章，包括 486 个常用 JavaScript 实例。每个实例都提供了代码分析及效果演示，可以帮助读者轻松掌握 JavaScript 的开发技巧，并从中找到网站开发的乐趣。

本书适用于初、中级 Web 开发人员，也是高级开发人员的查询宝典。对于 JavaScript 入门级读者来说，学习这些实例，可以更快地提高 JavaScript 的开发水平。对于中级 Web 开发人员而言，掌握这些实例后，可以开发出更加安全、快速、完善的 Web 2.0 网站。

本书提供了 Web 2.0 时代所必须掌握的一些技巧实例，是一本学习网络开发技术的随身手册。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

网页开发手记：486 个 JavaScript 网页特效详解 / 吴绍兴，韩峰，胡可编著. —北京：电子工业出版社，2011.10
ISBN 978-7-121-14446-2

I. ①网… II. ①吴… ②韩… ③胡… III. ①JAVA 语言—网页制作工具 IV. ①TP312②TP393.092

中国版本图书馆 CIP 数据核字（2011）第 172671 号

责任编辑：高洪霞

印 刷：北京东光印刷厂

装 订：三河市皇庄路通装订厂

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本：787×1092 1/16 印张：38 字数：935 千字

印 次：2011 年 10 月第 1 次印刷

印 数：4000 册 定价：69.80 元（含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zltz@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前言

为什么要学习 JavaScript

互联网已经遍布在我们周围，读书、购物、银行、生活服务等一切都能通过互联网中的网站解决。网站是由众多的网页组成的，我们可以通过这些网页与网站交互（如查询、搜索），这些网页就是动态网页，实现动态要感谢 JavaScript。

JavaScript 的出现使得网页和用户之间实现了一种实时、动态、交互的关系，使网页包含更多活跃的元素和更加精彩的内容，所以要熟悉网页开发，就必须要学会 JavaScript，它是专门为制作 Web 网页而量身定做的一种编程语言。JavaScript 短小精悍、简单易学，又是在客户机上执行的，所以大大提高了网页的浏览速度和交互能力。

要制作网页，就必须用 JavaScript，要制作精美流畅的网页，就必须学好 JavaScript！

如何学习 JavaScript 开发

如果说 JavaScript 很难学，那是因为你没有找对方法，按照下面的步骤学习，保证在 1 天内能把它应用得得心应手。比如有一个任务：要求网页中有一个登录按钮，只要用户按下回车键，就能触发该按钮，那么你可以这么操作：

- ① 按照光盘中的讲解，学会 JavaScript 程序的运行方法。这一步很简单，10 分钟就可以掌握。
- ② 用 IE 浏览器，打开光盘中的“源文件\C02\2.2 按回车调用登录按钮.htm”文件，将会启动一个网页，书中对该文件的功能有详细的讲解。
- ③ 用记事本打开该文件，可以看到其中的源代码，根据书中的注释，理解代码的含义。把它直接复制到自己的项目中。

OK！按本书只要 3 步就搞定这个任务了！如果有兴趣，可以看看书中的“难点剖析”，或者什么都不用看，在需要某个功能的时候，到书里面查找即可。

本书的优势

1. **配有学习论坛** <http://www.rzchina.net>，读者可以在上面讨论技术，笔者会及时回答读者提问，并提供各种技术文章，帮助读者提高开发水平。
2. **光盘中赠送所有实例源代码**，帮助你快速入门和实现所需效果。
3. **技术最新，与时俱进**：目前市场上的 JavaScript 相关书籍大都介绍了其包含的基本语法，还有网络刚兴起时的一些 JavaScript 特效。本书不仅包含这些内容，而且结合目前比较流行的

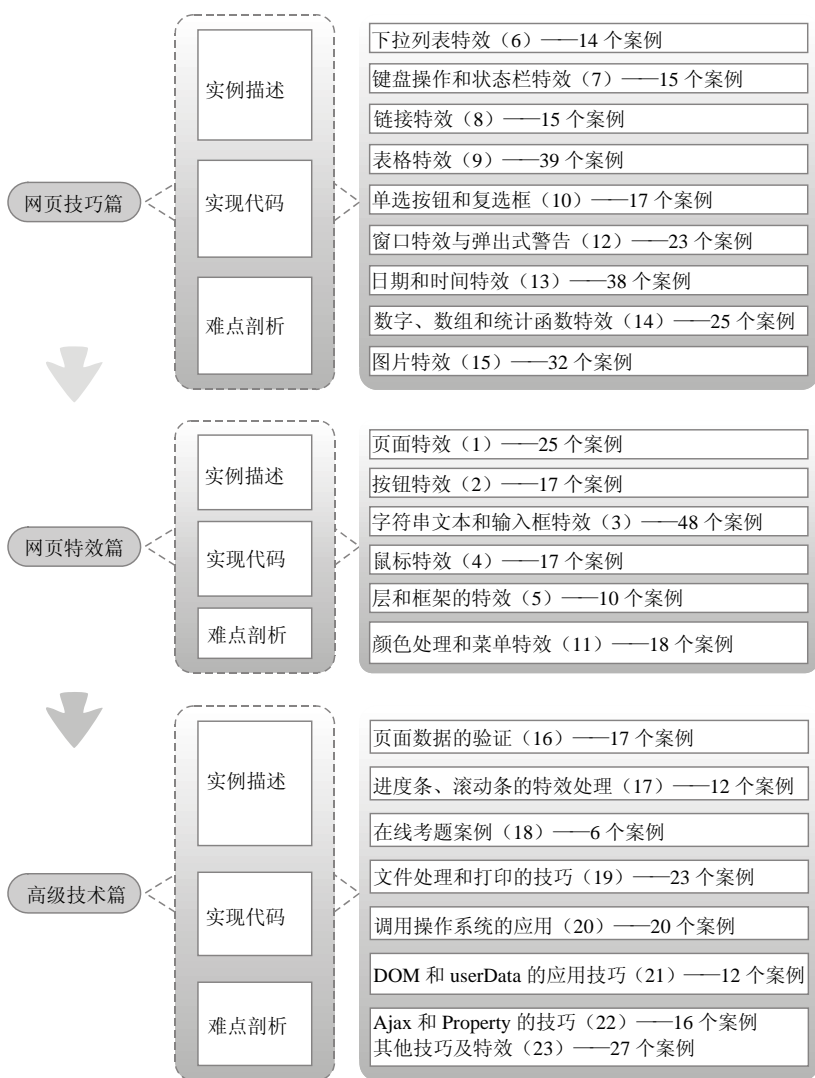
Ajax、DOM、Property 框架技术，提供了很多现在网络开发必须掌握的技巧实例。

4. **最全最流行的应用实例：**本书基本上涵盖了目前网络开发涉及的所有方向，从页面、文本、窗口、鼠标、日期时间等基本应用，到图像、滚动条、进度条、网络验证、文件处理、异步传输等深入应用。针对最流行的 Ajax 技术，提供了比较典型的实例。

5. **思路清晰，说明详细：**本书中每个实例都提供了 4 部分：实例的描述、实现的代码、实例的运行效果和实例的难点剖析。通过这 4 部分，读者可以轻松掌握每个实例的实现原理及 JavaScript 的应用方法。

6. **技巧难点，悉数掌握：**本书为每个实例都提供了难点剖析，用通俗易懂的语言，讲解 JavaScript 中的基本语法、对象及函数。读者可以轻松地解决应用中遇到的困难。

本书包括的内容



本书适合的读者

本书具有知识全面、实例精彩、指导性强的特点，力求以全面的知识及丰富的实例来指导读者掌握 JavaScript 网络应用技术。本书适合如下读者。

- Web 开发爱好者
- 流行网络技术爱好者
- 站长及网络维护人员
- 刚接触 B/S 应用的人员
- JavaScript 爱好者
- 网络开发人员
- 网页设计人员
- 正在做毕业设计的学生

本书作者

本书主要由南阳理工学院的吴绍兴、韩峰和胡可编写。其中第 1~11 章由吴绍兴编写，第 12~17 章由韩峰编写，第 18~23 章由胡可编写。

编者

目 录

第 1 章 页面特效.....	1
1.1 HTML 页面反向显示.....	2
1.2 页面自动最大化.....	2
1.3 页面自动刷新.....	3
1.4 页面的后退、刷新、前进.....	4
1.5 保护网页源代码.....	5
1.6 保护自己的网页不被放入框架.....	5
1.7 打印页面的出错原因.....	6
1.8 当前网页调用其他网页.....	7
1.9 倒计时载入页面.....	7
1.10 定义网页的关键字.....	8
1.11 进入页面同时弹出欢迎对话框.....	9
1.12 禁止网页另存.....	9
1.13 禁止页面加入缓存.....	10
1.14 离开页面时弹出对话框.....	11
1.15 判断页面是否添加了 W3C 声明.....	11
1.16 屏蔽网页中的脚本.....	12
1.17 以频道模式打开页面.....	12
1.18 自动识别网页类型.....	13
1.19 在网页中动态添加 Script 脚本.....	14
1.20 用 JavaScript 随机修改页面的标题.....	15
1.21 判断网页加载完毕.....	15
1.22 网页中嵌入播放器.....	16
1.23 设置指定网页为主页.....	17
1.24 使用 JavaScript 传递页面参数.....	18

1.25	冻结页面	19
第 2 章	按钮特效	20
2.1	页面刷新按钮	21
2.2	按 Enter 键调用登录按钮	21
2.3	动态创建按钮	22
2.4	平面按钮	23
2.5	按钮的嵌入效果	23
2.6	改变状态栏信息按钮	24
2.7	定义按钮的热键	25
2.8	取得控件的绝对位置	25
2.9	删除时的确认提示	26
2.10	按钮只能单击一次	27
2.11	防止按钮连击	27
2.12	图片式按钮	28
2.13	在按钮底部显示文字	29
2.14	选择不同的列表项时显示不同的按钮	30
2.15	使用按钮控制文本渐变	31
2.16	带翻页效果的公告栏	34
2.17	动态设置控件的事件	36
第 3 章	字符串文本和输入框特效	37
3.1	只带下画线的输入框	38
3.2	限定文本框可输入字符数	38
3.3	文字过长时的省略界面	39
3.4	输出 26 个英文字母	40
3.5	首字母变为大写	40
3.6	textarea 自适应文字行数	42
3.7	禁止文本的复制和粘贴	42
3.8	控制两个文本框只输入其一	43
3.9	判断编辑器中是否包含特殊字符	44
3.10	判断文本中回车车的数量	45
3.11	判断字符串中有多少汉字	45
3.12	去除字符串前后的空格	46
3.13	刷新时清空所有文本框	47
3.14	随意改变大小的文本框	48
3.15	文本框的自动全选	48



3.16	文本框滚动导航	49
3.17	按钮获取焦点	50
3.18	文本框获取焦点弹出下拉框	51
3.19	文本框简单的单击效果	52
3.20	文字的打字效果	53
3.21	文字滚动	54
3.22	文字滑动	55
3.23	文字跳动特效	57
3.24	荧光效果的文本	58
3.25	文字逐个闪亮——霓虹灯效果	59
3.26	旋转式的变色文字特效	61
3.27	《黑客帝国》中的字符下落效果	62
3.28	获取表单中文本框的个数	65
3.29	光标停在文本框最后	66
3.30	分行取 textarea 中的值	67
3.31	自动插入文本	68
3.32	选取 textarea 中的指定行	69
3.33	文本放大镜	70
3.34	文本框的默认输入法	71
3.35	文本框中显示网页中选中的内容	72
3.36	文字的垂直滚动	73
3.37	文字幻灯片	74
3.38	随机动态文字效果	75
3.39	实现 textarea 的自动滚动	76
3.40	使用 marquee 实现文字上下滚动	77
3.41	类似安装效果的 textarea 滚动	78
3.42	始终显示在最顶端的文本	79
3.43	JavaScript 过滤 SQL 注入字符	80
3.44	textarea 内实现行的翻页效果	81
3.45	textarea 中的文本插入	82
3.46	查找两段文本中相同的词句	83
3.47	自动保存网页的文本	85
3.48	文本编辑器	86

第 4 章 鼠标特效

4.1	禁用鼠标右键	94
4.2	使鼠标滚轮失效	94

4.3	状态栏显示鼠标位置	95
4.4	单击鼠标右键到指定页	96
4.5	鼠标放到图片上会显示另外一张图片	97
4.6	鼠标形状定义大全	98
4.7	鼠标移入移出时颜色变化	99
4.8	跟随鼠标的文字	99
4.9	跟随鼠标的彩色文字	101
4.10	跟随鼠标的魔法文字	103
4.11	跟随鼠标的星星	105
4.12	跟随鼠标的旋转背景	107
4.13	图片跟随鼠标	109
4.14	围绕鼠标的文本	110
4.15	鼠标旁边的提示信息	112
4.16	鼠标移到下拉框时自动全部打开	114
4.17	checkbox 鼠标移入移出的特效	115
第 5 章 层和框架的特效		117
<hr/>		
5.1	div 层提示效果	118
5.2	层自动滚动到底端	119
5.3	div 的自动滚动	120
5.4	div 的折叠效果	121
5.5	圆角 div	123
5.6	动态添加 iframe 框架	124
5.7	用层实现长篇文章分页	124
5.8	iframe 自适应高度	126
5.9	类似 MSN 的消息提示	128
5.10	只打印 iframe 的内容	131
第 6 章 下拉列表特效		132
<hr/>		
6.1	下拉列表框实现多选	133
6.2	实现两个 select 的同步	133
6.3	被选中的列表项下次不能再选	134
6.4	不带滚动条的 select	135
6.5	从一个下拉列表往另一个下拉列表添加内容	136
6.6	改变列表项的上下顺序	138
6.7	给下拉列表框数据分组	139
6.8	获取下拉列表框的选择	140



6.9 类 IE 下拉列表框	141
6.10 下拉列表框式邮件发送	142
6.11 手动调整的列表框	143
6.12 下拉框式网站导航	145
6.13 综合的搜索引擎	146
6.14 经典的 ListView 列表框	147

第 7 章 键盘操作和状态栏特效

7.1 按功能键返回首页	151
7.2 Enter 键实现 Tab 键功能	151
7.3 Ctrl+Enter 提交数据	152
7.4 IE 中屏蔽退格键 (BackSpace)	153
7.5 屏蔽键盘所有键	153
7.6 JavaScript 捕获方向键	154
7.7 状态栏变化信息	155
7.8 状态栏的跑马灯效果	157
7.9 状态栏缩放文字	158
7.10 状态栏文字来回显示	159
7.11 交替闪烁的状态栏	160
7.12 状态栏的分解显示文本特效	161
7.13 状态栏文字从右弹出	162
7.14 状态栏中文字从中间分开显示	164
7.15 屏蔽掉 IE 自带的功能键	165

第 8 章 链接特效

8.1 关闭窗口的链接	167
8.2 不用 CSS 实现链接样式的变化	167
8.3 让链接没有下画线	168
8.4 去掉超链接单击时的边框	169
8.5 提取页面中所有链接	169
8.6 一个链接打开两个地址	170
8.7 为链接提供下拉菜单	171
8.8 按钮链接	172
8.9 弹出鼠标所指的链接地址	172
8.10 链接的注释	173
8.11 为超链接同时绑定单击和双击事件	174
8.12 带链接的滚动字幕	175

8.13	会跳舞的链接	176
8.14	检测站点的链接速度	177
8.15	文本链接的渐变效果	179
第 9 章	表格特效	182
9.1	用 table 做的镜框	183
9.2	全自动单元格	183
9.3	突出的表格	184
9.4	让表格有提示信息	185
9.5	闪亮的表格边框	186
9.6	表格的宽度固定后内容自动换行	187
9.7	表格的排序	187
9.8	表格的斜线	189
9.9	table 中的文字滚动	191
9.10	JavaScript 遍历 table 的行和列	191
9.11	表格按 Enter 键自动生成新行	192
9.12	单击单元格背景变色	193
9.13	单击表格某行后其他行隐藏	194
9.14	单击表头实现表格排序	196
9.15	单击单元格显示行的详细信息	197
9.16	表格设置为“100%”时获取表格的宽度	198
9.17	表格选中后变色	199
9.18	表格中隐藏下级表格	201
9.19	表格自动下移	202
9.20	动态创建固定列数的表格	203
9.21	动态改变表格列宽	204
9.22	动态改变表格的行顺序	205
9.23	动态生成包含合并单元格的表格	207
9.24	用键盘上下键实现表格行的上下选择	208
9.25	用 JavaScript 隐藏或显示表格列	210
9.26	滚动的表格	212
9.27	交换表的行	213
9.28	动态拖动表格的宽度	214
9.29	可输入内容的表格	216
9.30	可以分级的表格隐藏	216
9.31	动态创建表格并实现分页	218
9.32	删除表格指定行	219



9.33	设置表格的交替行颜色	221
9.34	双击单元格变为可编辑	221
9.35	鼠标经过表格时列变色	222
9.36	鼠标选择表格中的多行	224
9.37	使用 JavaScript 向表格中写入数据	225
9.38	类 C# GridView 的编辑效果（一）	226
9.39	类 C# GridView 的编辑效果（二）	227
第 10 章	单选按钮和复选框	229

10.1	选择了哪一个单选按钮	230
10.2	单击文字实现单选按钮的选定	230
10.3	被选中的复选框求和	231
10.4	复选框组选	232
10.5	复选框分组全选	233
10.6	复选框和文本框的联动效果	235
10.7	单击任意单元格都能自动选中复选框	236
10.8	调用复选框后面的文字	237
10.9	两组复选框互斥问题	237
10.10	使用复选框控制文本框	239
10.11	选中表格行前的复选框则行变色	240
10.12	用 JavaScript 生成面包屑导航	241
10.13	复选框的反选	242
10.14	复选框全选（一）	243
10.15	复选框全选（二）	244
10.16	获取复选框的选择项	245
10.17	改变 select 选中项的颜色特效	246

第 11 章 颜色处理和菜单特效.....248

11.1	背景颜色测试	249
11.2	RGB 颜色在线转换	250
11.3	颜色切换板	251
11.4	下拉菜单	252
11.5	左键弹出式菜单	254
11.6	目录样式的下拉菜单	255
11.7	网页中的选项卡	256
11.8	静态导航菜单	258
11.9	烟花效果的下拉菜单	260

11.10	网络导航条	262
11.11	隐藏式菜单	264
11.12	仿 Flash 菜单	268
11.13	滚动导航菜单	270
11.14	幻灯片式的导航菜单	271
11.15	类似 QQ 主界面的菜单	273
11.16	三级联动菜单（一）	275
11.17	三级联动菜单（二）	277
11.18	树型目录菜单	280
第 12 章	窗口特效与弹出式警告	282
12.1	无关闭按钮的窗口	283
12.2	鼠标控制窗口开关	284
12.3	使窗口只在第一次访问时弹出	284
12.4	禁止弹出警告框	285
12.5	关闭窗口不提示的方法	286
12.6	关闭窗口时的提示	287
12.7	定时弹出窗口	287
12.8	调整窗口的大小	288
12.9	打开的窗口居中	288
12.10	打开窗口的等待提示	289
12.11	在打开的窗口中返回数据	290
12.12	创建弹出窗口	291
12.13	不允许窗口出现滚动条	292
12.14	页面打开的同时打开另外两个窗口	292
12.15	慢慢变大的窗口	293
12.16	设置新打开的窗口为活动窗口	294
12.17	页面随窗口的改变而改变	295
12.18	幻灯片式弹出窗口	295
12.19	弹出窗口生成器	297
12.20	关不掉的警告框	298
12.21	循环的警告框	299
12.22	屏蔽状态栏的错误提示	300
12.23	获取模式窗口的值	301
第 13 章	日期和时间特效	303
13.1	指定时间关闭页面	304



13.2 最简单的时间日期特效	304
13.3 获取时间的最简单方法	305
13.4 随日期变换的文本	305
13.5 输入框的默认值为当天	307
13.6 时间相加	308
13.7 12 小时制和 24 小时制的转换	308
13.8 标题栏显示时间	310
13.9 超过时间页面自动跳转	311
13.10 分时段问候用户	311
13.11 获取服务器时间	312
13.12 倒计时显示	313
13.13 背景时钟	314
13.14 计算某天是星期几	315
13.15 计算时间差	317
13.16 计算用户浏览网页的时间	318
13.17 记录页面的修改时间	318
13.18 将日期转换为字符串的方法	319
13.19 检测是否是闰年	320
13.20 年份加减函数	321
13.21 精确到千分之一秒	322
13.22 距离某天的时间	324
13.23 判断两个字符串日期的大小	326
13.24 显示登录时间	327
13.25 中文日期样式（一）	328
13.26 中文日期样式（二）	329
13.27 状态栏动态显示时间	330
13.28 页面访问时间限制	331
13.29 显示英文的上、下午时间标签	332
13.30 用 JavaScript 制作的特色时钟	333
13.31 自定义的日历	335
13.32 生日提醒器	339
13.33 时间的倒影	341
13.34 使用正则表达式验证日期	342
13.35 全面的日期选择功能	343
13.36 全球的时间查看表	345
13.37 无刷新定时取数据	347

13.38 取当月的最后一天	349
第 14 章 数字、数组和统计函数特效	351
14.1 边打字边显示字数	352
14.2 创建随机数	352
14.3 JavaScript 创建二维数组	353
14.4 截断小数点位数	355
14.5 删除数组中指定元素	355
14.6 数字选中后放大	356
14.7 统计字符数的方法	358
14.8 JavaScript 遍历数组	359
14.9 获取字符串型数组索引的数组长度	359
14.10 用 JavaScript 实现数组排序	360
14.11 数字千分位函数	361
14.12 读写 Cookie 的函数	362
14.13 获取 JavaScript 函数中的所有参数	364
14.14 奇偶数的判断	364
14.15 在 JavaScript 运行 VBScript 函数	365
14.16 购物篮中常用的计算总价效果	366
14.17 同一用户的来访统计	367
14.18 16 进制数转换为 10 进制数	369
14.19 将 URL 转化为 16 进制	370
14.20 小写金额转换为大写金额	370
14.21 通过两点坐标计算直线距离	372
14.22 随机抽取彩票	374
14.23 实时计算折扣	375
14.24 实用计算器	376
14.25 前面补 0 的方法	378
第 15 章 图片的特效	380
15.1 图片变形效果	381
15.2 图片的翻转效果	381
15.3 图片的模糊效果	382
15.4 图片的水印效果	383
15.5 图片淡出淡隐	384
15.6 图片的渐隐播放效果	385
15.7 文字环绕图片	386
15.8 图片切换的特殊效果	387



15.9	晃动的图片	389
15.10	定时消失的图片	391
15.11	QQ 图片一闪一闪的效果	392
15.12	设置 textarea 中的图片不处于编辑状态	392
15.13	禁止图片的复制	393
15.14	LOGO 像雪花一样落下	394
15.15	多幅图片分页滚动显示	397
15.16	循环滚动显示图片	399
15.17	图片的选择展示	403
15.18	图片新闻切换效果	404
15.19	判断上传图片的大小	405
15.20	上传图片时预览	406
15.21	对联广告	407
15.22	带关闭的对联广告	409
15.23	到边界反弹的漂浮图片	410
15.24	用键盘控制图片移动	413
15.25	预装载图片提高站点速度	414
15.26	始终在屏幕右下角的图片	415
15.27	可拖动的图片	416
15.28	等比例缩略图	418
15.29	用 JavaScript 导出图片到 Excel	419
15.30	使用 VML 打造可改变大小的圆框	420
15.31	JavaScript 实现文档结构图	421
15.32	判断一幅图片是否加载完毕	427

第 16 章 页面数据的验证.....428

16.1	验证字符串是否全由数字组成	429
16.2	验证表单项必须填写	429
16.3	判断用户输入是否为中文	430
16.4	验证列表框中的值是否重复	431
16.5	检测输入框的统一方法	432
16.6	E-mail 的验证	433
16.7	不使用正则验证 IP 地址	434
16.8	IP 地址输入框	434
16.9	判断变量是否已经定义	435
16.10	判断方法是否已经定义	436
16.11	表单验证样式	437

16.12	判断表单是否已修改	438
16.13	判断控件的类型	439
16.14	密码强度检查	440
16.15	身份证号的验证	442
16.16	JavaScript 生成验证码（一）	443
16.17	JavaScript 生成验证码（二）	444
第 17 章 进度条、滚动条的特效处理		446
<hr/>		
17.1	使用符号制作的进度条	447
17.2	用 table 制作进度条	448
17.3	用 CSS+JS 制作进度条（一）	449
17.4	用 CSS+JS 制作进度条（二）	450
17.5	进度条形式的下载效果	452
17.6	滑动条（一）	454
17.7	滑动条（二）	458
17.8	窗体滚动条随文字增加自动滚动	461
17.9	为 textarea 加横向滚动条	462
17.10	记录滚动条位置	463
17.11	彩色滚动条	464
17.12	Windows XP 的滚动条	464
第 18 章 在线考题案例		466
<hr/>		
18.1	在线考试代码（一）	467
18.2	在线考试代码（二）	468
18.3	在线测试代码	470
18.4	多选考试题	471
18.5	在线心理测试脚本	472
18.6	电脑检测健康情况	474
第 19 章 文件处理和打印的技巧		476
<hr/>		
19.1	判断上传文件的类型	477
19.2	改变上传文件控件的样式	477
19.3	上传文件一次完成	478
19.4	使用正则表达式判断文件扩展名	479
19.5	多附件上传效果	479
19.6	上传控件内容清空	481
19.7	textarea 显示记事本文件的内容	482



19.8	使用 FSO 读写文本文件.....	483
19.9	自动启动文件下载	485
19.10	创建 Excel 文件	485
19.11	JavaScript 导出数据到 Excel.....	486
19.12	JavaScript 读取自身文件内的 XML	488
19.13	将 XML 文件绑定到 table.....	489
19.14	使用 JavaScript 加载 XML 文件	490
19.15	动态加载 JavaScript 文件	492
19.16	防止 JavaScript 文件被其他站直接引用	493
19.17	检查机器是否安装 Word.....	493
19.18	打印当前页	494
19.19	打印预览	495
19.20	隐藏不想打印的页面内容	496
19.21	使用 ExecWB 直接打印	498
19.22	动态绑定 XML 文件	498
19.23	Kill Excel 的进程	499

第 20 章 调用操作系统的应用

20.1	JavaScript 操作剪贴板.....	502
20.2	打开硬盘驱动器	503
20.3	单击加入收藏夹	503
20.4	复制标题和网址	504
20.5	关闭输入法	505
20.6	检测屏幕分辨率.....	505
20.7	检测系统信息	506
20.8	显示本地计算机信息	507
20.9	检测浏览器浏览过的站点数	507
20.10	IE 文件菜单中的“打开”命令.....	508
20.11	打开“Internet 选项”对话框	509
20.12	打开 Windows 系统的画板	509
20.13	弹出保存对话框	510
20.14	进入页面弹出收藏夹	511
20.15	执行客户端的可执行程序	513
20.16	自动调用 OutLook 发送邮件	514
20.17	弹出窗口选择颜色	515
20.18	弹出框式邮件发送	516
20.19	把网站作为用户的 Active 桌面.....	517

20.20	判断是否安装了 Flash 插件	518
第 21 章	流行技术：DOM 和 userData 的应用技巧	520
21.1	使用 userData 保存文本内容	521
21.2	使用 userData 保存 select 标记	522
21.3	使用 userData 保存 checkbox 标记	523
21.4	使用 DOM 实现控件的替换	524
21.5	使用 DOM 实现控件的复制	525
21.6	使用 DOM 判断页面中控件是否嵌套	526
21.7	使用 DOM 获取页面中某控件的属性	526
21.8	将某行排在表格的最后	527
21.9	动态删除页面中的元素	528
21.10	克隆表格	529
21.11	拖动表格行改变顺序	530
21.12	表格操作常用方法	533
第 22 章	流行应用：Ajax 和 Property 的技巧	535
22.1	关机特效（一）	536
22.2	关机特效（二）	537
22.3	评星效果	538
22.4	输入框自动完成功能	539
22.5	Ajax 效果的字符串过滤	540
22.6	GMail 右上角的 Loading 效果	543
22.7	使用 XMLHTTP 获取天气预报	543
22.8	拖曳任意对象	545
22.9	避免打开无效页面	547
22.10	用 JavaScript 调用 Google AdSense	547
22.11	Ajax 效果——可拖曳的表格	548
22.12	JavaScript 调用 Web Service	552
22.13	用 JavaScript 实现编码解码	554
22.14	创建带属性的对象	555
22.15	用 prototype 实现 JavaScript 的继承	556
22.16	JavaScript 制作哈希表	556
第 23 章	其他技巧及特效	558
23.1	最简单的漂移特效	559
23.2	JavaScript 遍历对象中的所有属性	559



23.3	QQ 在线客服	560
23.4	查看网站的排名	561
23.5	定义全局变量	561
23.6	动态生成金字塔效果	562
23.7	动态修改 CSS 的样式	563
23.8	根据浏览器不同设置 CSS	563
23.9	汉字按拼音排序	564
23.10	划词搜索	565
23.11	加载大量 input 控件的快速方法	568
23.12	简繁体转换	569
23.13	将 HTML 转换为 JavaScript 脚本	570
23.14	脚本永不出错	571
23.16	浏览器毁灭者	572
23.17	罗列对象的属性和值	573
23.18	密码保护页	574
23.19	全角转半角	575
23.20	全屏广告	576
23.21	5 秒钟后消失的广告	576
23.22	输入的英文自动全大写	578
23.23	特殊扩散效果	578
23.24	提交信息等待界面	580
23.25	同时调用两个方法	581
23.26	自定义错误处理样式	582
23.27	FTP 网站登录	582

第 1 章 页面特效

本章导读

本章从 Web 开发最基础的页面开始，详细介绍常见的页面特效，并逐个分析这些特效的实现原理。本章还包含一些常用的网页应用技巧，如检查页面类型、加密页面源代码等。



1.1 HTML 页面反向显示

【实例描述】

默认的 HTML 页面内容是从左到右显示的。本例将学习如何将其设置为从右到左显示。

【实现代码】

```
<html dir=rtl>
<head>
<title>标题页</title>
<body>
这是一段伟大的历史<br />
来自全世界的瞩目和掌声
</body>
</html>
```

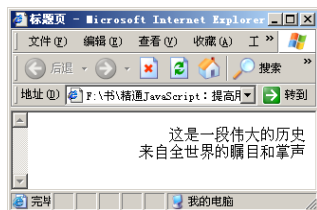


图 1-1 反向显示的效果

【运行效果】

反向显示的效果如图 1-1 所示。

【难点剖析】

本例的重点是 dir 属性，很多文本显示控件都具备这个属性，其用来控制文本的显示方向。“rtl”就是 right to left 的意思，表示从右到左显示。如果从左到右显示，就是“ltr”。

1.2 页面自动最大化

【实例描述】

一些内容比较多的网站为了方便用户的操作，通常在打开页面时就将窗口设置为最大化。本例介绍如何使页面打开时默认为最大化状态。

【实现代码】

```
<script language="javascript">
//定位左上角
self.moveTo(0,0);
//调整屏幕
self.resizeTo(screen.availWidth,screen.availHeight);
</script>
```

【运行效果】

最大化的页面效果如图 1-2 所示。

【难点剖析】

本例的重点是 screen 对象，用其控制窗口，并实现窗口的最大化，其中“availWidth”表示

屏幕允许的最大宽度，“availHeight”表示屏幕允许的最大高度。



图 1-2 最大化了的页面效果

1.3 页面自动刷新

【实例描述】

为了及时反映站点数据的变化，通常需要页面进行自动刷新。本例学习如何实现页面的自动刷新，刷新结果可以是当前页面，也可以转换到指定页面。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <meta HTTP-EQUIV="refresh" CONTENT="3;URL=http://www.google.com">
</head>
<body>
  <table id="mytbl" width="300" height="50" border="0" cellspacing="2" cellpadding="0"
  bgcolor="#FFb609">
    <tr>
      <td> 第一行第一列</td><td> 第一行第二列</td>
    </tr>
    <tr>
      <td> 第二行第一列</td><td> 第二行第二列</td>
    </tr>
  </table>
</body>
</html>
```

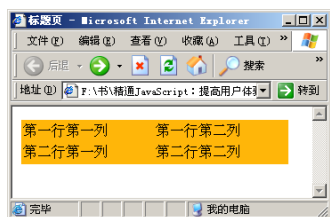


图 1-3 页面自动刷新实例的运行效果

【运行效果】

页面自动刷新实例的运行效果如图 1-3 所示。

【难点剖析】

本例的重点是 meta 元素，其属性“HTTP-EQUIV”设置为“refresh”时，会自动刷新当前页面，此属性包含两个重要的设置：CONTENT 和 URL，CONTENT 表示自动刷新的时间间隔，URL 表示刷新后的页面地址。

1.4 页面的后退、刷新、前进

【实例描述】

后退、刷新和前进是页面浏览时最常用的操作，Windows 自带的 IE 浏览器可以完成这 3 个功能，在 Web 应用系统中，有时需要屏蔽浏览器的工具栏，此时需要使用代码实现页面的这 3 个操作。本例学习如何使用代码完成页面的后退、刷新和前进。

【实现代码】

```
<script language="javascript">
function back()
{
    history.go(-1); //后退 1 页
}
function forward()
{
    history.go(+1); //前进 1 页
}
function refresh()
{
    history.go(-0) //刷新
}
</script>
```

需要在 body 中添加 3 个按钮，分别调用上面的 3 个方法，代码如下所示。

```
<input type="button" value="后退" onclick="back()">
<input type="button" value="刷新" onclick="refresh()">
<input type="button" value="前进" onclick="forward()">
```

【运行效果】

后退、刷新、前进操作按钮的运行界面如图 1-4 所示。

【难点剖析】

本例的重点是“history”对象，其用来存储浏览器的历史记录，其参数为正数时，表示前进页面，负数则表示后退页面，如要后退 2 页则使用“history.go(-2)”。

1.5 保护网页源代码

【实例描述】

大部分网页的源代码可以使用 IE 提供的“查看源代码”命令查看。本例学习一种方法，可以屏蔽对网页源代码的查看。

【实现代码】

```
<html>
<head>
<title>测试是否能看到源代码</title>
<script>
function clear()
{
    Source=document.body.innerHTML;           //获取文档的原有内容
    document.open();                           //打开文档
    document.write("代码已经被屏蔽");         //输出提示内容
    document.close();                          //关闭文档
    document.title="看不到源代码";            //文档标题
    document.body.innerHTML=Source;           //重新写入旧内容
}
</script>
</head>
<body onload=clear()>
<marquee>测试下能否看到源码</marquee>
</body>
</html>
```

【运行效果】

使用“查看源代码”命令后的效果如图 1-5 所示。

【难点剖析】

本例的原理是首先将文档的内容保存在一个变量中,然后清空文档的内容,最后再在文档中显示旧内容。其实文档内容显示的就是“document.write”输出的内容。

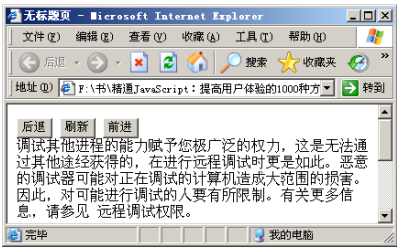


图 1-4 后退、刷新、前进操作按钮的运行界面

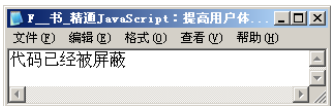


图 1-5 使用“查看源代码”命令后的效果

1.6 保护自己的网页不被放入框架

【实例描述】

随着网络信息的传播，很多网站不再具备自己的内容，而是使用框架，加载其他网站的内容。如果不允许网页被他人加载，可使用本例提供的方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```



```
<head>
<title>标题页</title>
<Script LANGUAGE="JavaScript">
if(self!=top){
top.location=self.location;           //判断是否是顶层，不是则将当前页设置为顶层
}
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是对层的判断。在网页中，可以被框架加载的页面不能位于“top”层，所以通过判断当前页面是否是“top”层，来判断是否能被框架加载。如果此页面不位于“top”层，则使用“top.location”将其设置为顶层，可使其不会被加载。

1.7 打印页面的出错原因

【实例描述】

很多人习惯手写 JavaScript 代码，而不使用 JavaScript 编辑器。这样操作的缺点在于不便于错误的调试，本例提供一种打印错误的方法，让调试人员可以轻松地找到错误所在。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function getError()
{
    try
    {
        x =new test;           //产生错误
    }
    catch(e)
    {
        document.write(e.description)    //打印错误结果
    }
}
</script>
</head>
<body onload="getError()" >
</body>
</html>
```

【难点剖析】

本例的重点是“try...catch”语句的运用。try 语句用来运行代码，当代码有错误发生时，则转到 catch 语句继续执行。在 catch 语句中，使用“e.description”获取错误信息的描述，然后通过“document.write”方法，将错误信息显示在页面上。

1.8 当前网页调用其他网页

【实例描述】

一个页面可以使用 iframe 来加载另一个页面，但有些 IE 是禁止使用框架的，所以本例使用另外一个方法，实现在当前页中调用另一个网页。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<object type="text/x-scriptlet" width="150" height="100" data="http://www.baidu.com">
</object>
</body>
</html>
```

【运行效果】

调用其他页面后的效果如图 1-6 所示。

【难点剖析】

本例的重点是 object 标签的使用。其 type 属性是“text/x-scriptlet”，其中“x-scriptlet”是 IE 的一个插件。data 属性表示要加载的数据。



图 1-6 调用其他页面后的效果

1.9 倒计时载入页面

【实例描述】

倒计时是常用的载入页面时的等待方式，本例学习如何计算倒计时的时间，同时学习自动加载页面的相关设置。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <meta http-equiv="refresh" content="11; url=http://www.google.com">
  <script language="JavaScript">
    // 获取当前时间
```



```
startday = new Date();
clockStart = startday.getTime();

function initStopwatch()
{
    var myTime = new Date();
    var timeNow = myTime.getTime();
    var timeDiff = timeNow - clockStart;    //获取间隔时间
    this.diffSecs = timeDiff/1000;         //因为时间以毫秒为单位
    return(this.diffSecs);                 //返回间隔秒数
}

function getSecs()
{
    var mySecs = initStopwatch();
    var mySecs1 = ""+mySecs;
    //以倒计时方式显示时间
    mySecs1= 10 - eval(mySecs1.substring(0,mySecs1.indexOf("."))) + "秒";
    document.form1.timespent.value = mySecs1;
    window.setTimeout('getSecs()',1000);
}
</script>
</head>
<body bgcolor="#ffffff" onLoad="window.setTimeout('getSecs()',1)">
    <center>
        10 秒后将加载页面:
        <form name=form1><input size=9 name=timespent></form>
    </center>
</body>
</html>
```

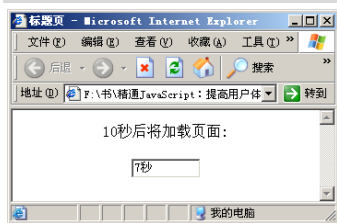


图 1-7 倒计时等待效果

【运行效果】

倒计时等待效果如图 1-7 所示。

【难点剖析】

本例的重点是加载页面、计算剩下的时间。加载页面使用的是 Meta 元素，其包含一个属性 Content，它包含两个参数：需要等待的时间和要加载的页面地址。计算剩余时间使用了两个日期相减的方式，注意相减的结果是毫秒数。

1.10 定义网页的关键字

【实例描述】

在网页中加入关键字，可以供某些搜索引擎使用，其会利用该关键字为网站做索引，这样当用户在搜索网站中输入相同的关键字时，网站就会被搜索到。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```



```
<head>
<title>标题页</title>
<meta name="Keywords" content="net,enterprise,ERP,c#">
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是 meta 元素，其中有个属性“content”用来设置网站包含的关键字，如果要提高网站的排名，还可以设置重复的关键字。

1.11 进入页面同时弹出欢迎对话框

【实例描述】

当用户进入网站时，会先弹出一个对话框，说明此网站的作用并欢迎用户。此功能一般用在需要特殊说明的网站，如禁止非法拷贝等声明。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<SCRIPT LANGUAGE="JavaScript">
    alert("欢迎您进入在线考试系统，答题期间请不要打电话！")
</script>
</body>
</html>
```

【运行效果】

弹出对话框的效果如图 1-8 所示。

【难点剖析】

本例的重点是 body 中的内容。如果需要在页面开始的时候执行某些操作，一般将方法绑定到 body 的“onload”事件中。本例并没有这么做，而是利用 JavaScript 逐行解释执行的原理，将 JavaScript 代码直接放在 body 内部。

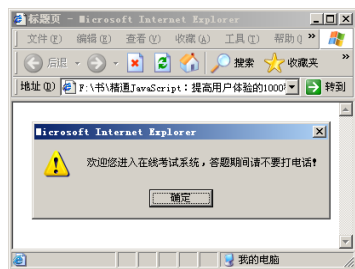


图 1-8 弹出对话框的效果

1.12 禁止网页另存

【实例描述】

很多网站为了保护自己的数据资料，禁止用户将网页另存到本地。本例学习如何禁止另存行为。



【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
</head>
<body>
<noscript><iframe src=*.html></iframe></noscript>
</body>
</html>
```

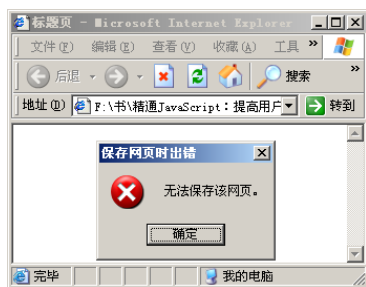


图 1-9 禁止网页另存的提示

【运行效果】

禁止网页另存的提示如图 1-9 所示。

【难点剖析】

本例中有两个重点：noscript 元素和框架。noscript 是 HTML 的元素，如果浏览器支持 JavaScript，则会忽略此元素内的代码，如果不支持 JavaScript，则会显示 noscript 内的代码。iframe 表示框架，其数据源是所有的 HTML 文件。noscript 和 iframe 结合就禁止了网页的另存行为。

1.13 禁止页面加入缓存

【实例描述】

默认情况下，用户浏览过的页面一般都会加入缓存，即使断网后也能继续浏览。本例将学习如何禁止页面加入缓存。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<body>
<input type=text name="txt1" value="this is test!">
<input type=button value="转换文本" onClick="javascript:changeCase(txt1)">
</body>
</html>
```

【难点剖析】

本例的重点是 HTML 的 META 元素。META 元素包含两个重要的属性：“HTTP-EQUIV”和“CONTENT”。“HTTP-EQUIV”相当于 HTTP 的文件头，其可以向浏览器传回一些有用信息，以帮助正确和精确地显示网页内容，与之对应的属性为“CONTENT”，“CONTENT”中的内容其实就是各个参数的值。

1.14 离开页面时弹出对话框

【实例描述】

为了体现网站对用户的关怀，可在用户离开页面时给予一些提示，如“谢谢使用，购买的东西不要忘记付款!”。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body bgcolor="#fef4d9" onUnload="window.alert('谢谢你的光临!欢迎下次再来!')">
</body>
</html>
```

【难点剖析】

本例的重点是 body 标签的“onUnload”事件。“onUnload”是离开页面时触发的事件。一般用在 body 或 frameset 标签上，如<body onUnload=" ">。在带框架集和框架的分层页面中，单个框架的 onUnload 事件放置在每个框架文件的 body 标签上，要比框架集的 onUnload 事件早发生。

1.15 判断页面是否添加了 W3C 声明

【实例描述】

当文档中添加了 W3C 的声明后，表示文档结构和对象都遵循 W3C 的规定，此时对象的一些调用方法可能会改变，如窗体的高度应使用“document.documentElement.clientHeight”，而不一定是“document.body.clientHeight”。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
if ( document.compatMode=="CSS1Compat" )
alert("本文档添加了 W3C 的声明")
else
alert("本文档未添加 W3C 的声明")
</script>
</head>
<body>
</body>
</html>
```

**【难点剖析】**

本例的重点是“document.compatMode”变量的值。当文档有了标准声明时，“document.compatMode”的值就等于“CSS1compat”，因此，可以根据“document.compatMode”的值来判断文档是否加了标准声明。

1.16 屏蔽网页中的脚本

【实例描述】

为了网页的显示需要，有时候需要将某段暂时不需要的脚本屏蔽掉。本例学习如何屏蔽网页中的 JavaScript 脚本。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<noscript>
<script LANGUAGE="JavaScript">
    alert("看看能不能提示");
</script>
</noscript>
</body>
</html>
```

【难点剖析】

本例的重点是 noscript 元素。将不需要的脚本放在<noscript>和</noscript>元素之间，可以屏蔽掉这些脚本的运行。

1.17 以频道模式打开页面

【实例描述】

在打开一些警告或说明的页面时，通常都不允许某些操作，如最大化、最小化等。本例介绍如何打开一个只有标题栏和关闭按钮的界面，这种模式被称为频道模式。

【实现代码】

```
<script language="javascript">
function channel()
{
    window.open("以频道模式打开页面 2.htm", "_blank", "channelmode")
}
</script>
```

需要创建一个名为“以频道模式打开页面 2.htm”的页面，内容不限。

【运行效果】

打开的频道模式页面如图 1-10 所示。

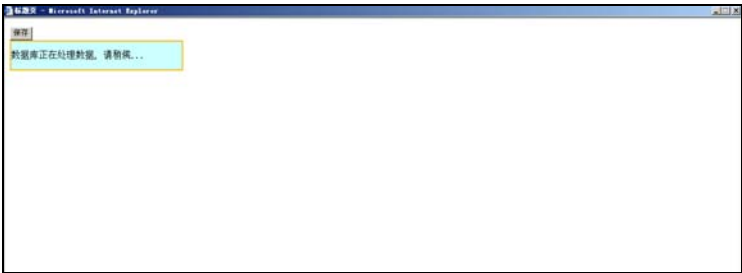


图 1-10 打开的频道模式页面

【难点剖析】

本例使用了“window.open”方法，用来打开一个新的页面，同时指定打开的参数“channelmode”，表示以频道模式打开新页面。

1.18 自动识别网页类型

【实例描述】

随着网站开发技术的更新，网页类型变得越来越丰富。本例将介绍如何自动识别网页的类型。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
    URL = window.location.href;           //获取网页当前 URL 地址
    ishtm = (URL.indexOf('.htm') > -1);    //判断地址中包含的字符串
    ishtml = (URL.indexOf('.html') > -1);
    isshtml = (URL.indexOf('.shtml') > -1);
    isphtml = (URL.indexOf('.phtml') > -1);
    isaspx = (URL.indexOf('.aspx') > -1);
</script>
</head>
<body>
<SCRIPT LANGUAGE="JavaScript">
//一个全局变量，用来判断网页类型
if (isphtml)
document.write("这是一个.phtml 文件!");
else if (isshtml)
document.write("这是一个.shtml 文件!");
else if (ishtml)
```

```
document.write("这是一个.html 文件!");  
else if (ishtm)  
document.write("这是一个.htm 文件!");  
else if (aspx)  
document.write("这是一个.aspx 文件!");  
else {  
document.write("无法识别该类型文件.");  
}  
</script>  
</body>  
</html>
```

【难点剖析】

本例的重点是字符串的“indexOf”方法。代码中首先通过“window.location.href”获取当前网页的 URL 地址，然后使用“indexOf”方法判断地址中包含的字符串。

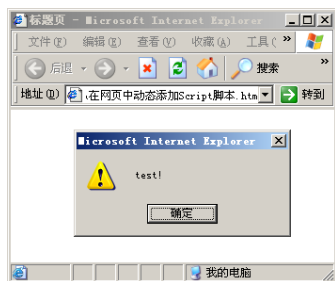
1.19 在网页中动态添加 Script 脚本

【实例描述】

Script 脚本可以在页面中静态地设置好，也可以使用“src”属性调用页面外部的脚本文件。本例学习一个方法，可以在页面运行过程中，动态添加 Script 脚本。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
<script>  
    o=document.createElement("script");           //动态创建 script 元素  
    o.text="alert('test!')";                         //弹出提示信息  
    document.documentElement.childNodes[0].appendChild(o); //将脚本添加到页面中  
</script>  
</head>  
<body>  
</body>  
</html>
```



【运行效果】

本例的运行效果如图 1-11 所示。

【难点剖析】

本例使用“createElement”方法，创建了一个“script”元素，并设置元素的内容为“alert('test!')”，最后使用“appendChild”方法，将此元素添加到窗体中。

图 1-11 本例的运行效果

1.20 用 JavaScript 随机修改页面的标题

【实例描述】

为了显示页面的动态效果，可设置页面的标题为多个，并动态轮换显示。本例通过随机数和数组，实现这一效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="javascript">
//随机显示 TITLE
var til = new Array(5);
til[0] = '第一个标题'
til[1] = '第二个标题'
til[2] = '第三个标题'
til[3] = '第四个标题'
til[4] = '第五个标题'
var indextitle = Math.floor(Math.random() * til.length); //获取一个随机整数
onload=function chaTitle() {
    var curTitle=til[indextitle]; //获取数组中的指定项
    window.document.title+=" - "+curTitle; //更改当前页的标题
}
</script>
</body>
</html>
```

【难点剖析】

本例将改变标题的代码写在了“onload”事件中，所以只有在加载页面的时候，标题才会改变。可按 F5 不断刷新页面，测试标题栏的随机标题。

1.21 判断网页加载完毕

【实例描述】

有时候需要在某个页面加载完毕后，执行一些操作或进行一些判断。如何判断页面加载完毕呢？本例通过一个“iframe”框架，学习如何判断页面加载完毕。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
```



```
<title>标题页</title>
</head>
<body>
<iframe onload=alert("加载完毕") src='http://www.sina.com.cn'></iframe></body>
</html>
```

【运行效果】

判断网页加载完毕的效果如图 1-12 所示。

【难点剖析】

iframe 框架是在当前页面中嵌入另一个页面。“onload”事件是页面加载完成后触发的事件。所以使用“框架+onload”的方法，可以轻松判断页面是否加载完毕。

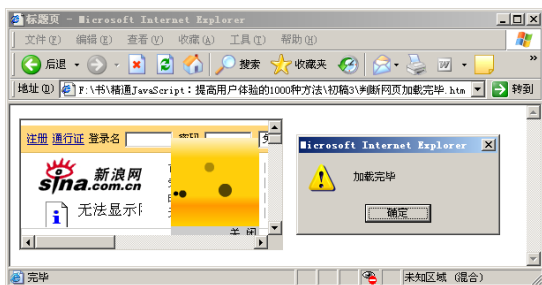


图 1-12 判断网页加载完毕的效果

1.22 网页中嵌入播放器

【实例描述】

很多网站提供视频教程，这些教程可以使用 MediaPlayer 播放，也可以用 RealOne 播放。本例学习如何将 MediaPlayer 播放器嵌入到网页中。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<OBJECT ID="play" WIDTH=450 HEIGHT=150 CLASSID="CLSID:6BF52A52-394A-11D3 -B153-
00C04F79FAA6">
<param name="URL" value="http://go.microsoft.com/fwlink/?LinkID=91175">
<param name="rate" value="1">
<param name="balance" value="0">
<param name="currentPosition" value="0">
<param name="defaultFrame" value>
<param name="playCount" value="1000">
<param name="autoStart" value="0">
<param name="currentMarker" value="0">
```



```

<param name="invokeURLs" value="-1">
<param name="baseURL" value>
<param name="volume" value="100">
<param name="mute" value="0">
<param name="uiMode" value="full">
<param name="stretchToFit" value="0">
<param name="windowlessVideo" value="0">
<param name="enabled" value="-1">
<param name="enableContextMenu" value="-1">
<param name="fullScreen" value="0">
<param name="SAMIStyle" value>
<param name="SAMILang" value>
<param name="SAMIFilename" value>
<param name="captioningID" value>
</OBJECT>
</body>
</html>

```

【运行效果】

嵌入到网页中的播放器如图 1-13 所示。

【难点剖析】

本例的重点是 OBJECT 组件的使用。Microsoft 为了扩展浏览器的功能，提供了多个组件，这些组件都通过 OBJECT 调用，调用时需要指定组件的“ID”，同时还要指名其“CLASSID”属性，其他参数可选，用来设置 MediaPlayer 的一些通用属性。

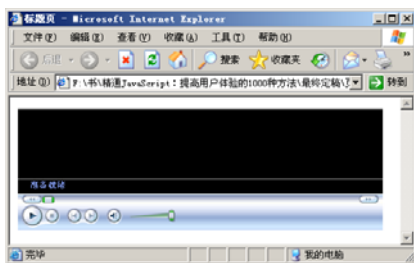


图 1-13 嵌入到网页中的播放器

1.23 设置指定网页为主页

【实例描述】

为了方便用户以后访问当前网站，可以设置一个链接，让用户方便地将网页设为首页。本例就学习如何设置浏览器的主页。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<a href="#" onClick="this.style.behavior='url(#default#homepage)';this.
setHomePage('http://www.google.com');">设为首页
</a></body>
</html>

```



【难点剖析】

本例的重点是“behavior”属性。此属性是设置或检索对象的 DHTML 行为。使用语法如下所示：

```
behavior : url (url) | url (#objID ) | url (#default#behaviorName)
```

本例的使用方法是“url(#default#homepage)”，用来设置 IE 的默认行为，“homepage”表示主页。

1.24 使用 JavaScript 传递页面参数

【实例描述】

ASP 和 ASP.NET 都提供获取页面参数的对象，而 JavaScript 只能通过自定义方法实现。本例将学习如何获取页面传递过来的参数。

【实现代码】

```
<script LANGUAGE="JavaScript">
function GetArgs(parms, parmName)
{
    var argIndex=parms.indexOf("?")           //获取?所在的位置
    var arg=parms.substring(argIndex+1);      //获取?后面的字符串
    var valArg = "";
    args = arg.split("&");                    //使用 split 将参数分解为数组
    for(var i = 0; i < args.length; i ++){
        str = args[i];
        var arg = str.split("=");
        if(arg.length <= 1) continue;         //没有参数的时候
        if(arg[0] == parmName)
            valArg = arg[1];                  //获取指定参数的值
    }
    return valArg ;
}
function PageParm()
{
    var myname=GetArgs(window.location.href, "name");
    document.write("您请求的参数值为: "+myname);
}
</script>
```

【运行效果】

带传递参数的页面效果如图 1-14 所示。

【难点剖析】

本例的重点是浏览器中参数传递的格式。从一个页面传递参数到另一个页面的格式为“页

面地址？参数 1 名=参数值&参数 2 名=参数值”，其中“？”表示后面紧跟的字符串是参数，“&”用来连接两个参数。获取页面地址使用的是“window.location.href”。获取页面地址后使用“string”对象的一些方法分解字符串，以实现参数的提取。



图 1-14 带传递参数的页面效果

1.25 冻结页面

【实例描述】

页面被冻结后，鼠标单击页面中的任何按钮，都不会有反应。如果要关闭页面，只能通过“Ctrl+Alt+Del”组合键。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script>
function freeze()
{
    alert ('对不起，死机了!!!');           //首先提示已经完成效果
    while(true){                           //进入循环
        window.history.back(-1)           //后退一格
    }
}
</script>
<FORM>
<INPUT TYPE="BUTTON" VALUE="Freeze" onClick="freeze()">
</FORM>
</body>
</html>
```

【难点剖析】

本例通过“freeze”方法，实现一个页面的冻结效果。其中“freeze”方法中，使用了一个“while(true)”循环，此循环的条件一直为真，即这是一个无法结束的循环。

第 2 章 按钮特效

本章导读

按钮是客户端用来与服务器交互的主要控件。本章将介绍常见的按钮特效，如单击、连击等，并学习如何在客户端动态生成按钮、如何定义按钮的热键等。

2.1 页面刷新按钮

【实例描述】

页面的 meta 元素可以实现自动刷新功能,但如果要在程序执行中需要刷新页面,则不能使用此元素。本例通过设置一个刷新按钮,学习如何在程序中设置刷新代码。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function renovates(){
    document.location.reload();
}
</script>
</head>
<body onload="alert('页面刷新了!');">
<p align="center"><input type="button" name="renovates" value="刷新" onclick=
"renovates()" /></p>
</body>
</html>
```

本例只有一个按钮,所以没有给出图示,读者可以根据代码亲自试验。

【难点剖析】

本例的难点是页面的刷新方法“reload”,如果要使用 meta 元素,则使用 refresh 属性,而在程序中,则使用“document.location”的方法“reload”实现页面重新载入的效果。

2.2 按 Enter 键调用登录按钮

【实例描述】

为了方便用户操作,在登录邮箱或论坛时,如果用户输入了用户名和密码,按 Enter 键时,都会自动调用登录按钮。本例学习如何实现此功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Check Score</title>
<script language="JavaScript">
function keyLogin(){
    if (event.keyCode==13)                //按 Enter 键的键值为 13
        document.getElementById("input1").click();    //调用登录按钮的登录事件
}
</script>
```



```
</head>
<body onkeydown="keyLogin();">
<input id="input1" value="登录" type="button" onclick="alert('调用成功!')">
</body>
</body>
</html>
```

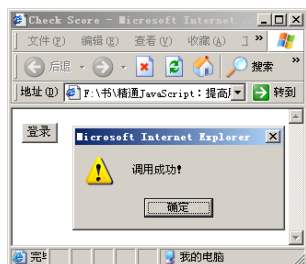


图 2-1 按 Enter 键后的效果

【运行效果】

按 Enter 键后的效果如图 2-1 所示。

【难点剖析】

本例的重点是如何判断用户敲击的是 Enter 键,这通过判断键值“keyCode”实现,“Enter”的键值为 13。使用“getElementById”方法,通过按钮的 ID 找到页面中的“登录”按钮,然后直接调用其“click”方法即可。

2.3 动态创建按钮

【实例描述】

为了提高网页的访问速度,可使用 JavaScript 创建控件,而不需要向服务器重新提交一次页面数据。本例通过 JavaScript 的 DOM 对象,实现按钮的动态创建。

【实现代码】

```
<html>
<head>
<title>动态创建按钮</title>
<script language="javascript">
var i=0 ;
function addInput()
{
var o = document.createElement("input");           //使用 DOM 的创建元素方法
o.type = "button" ;                                //设置元素的类型
o.value = "按钮" + i++ ;                             //设置元素的值
o.attachEvent("onclick",addInput);                  //为控件添加事件
document.body.appendChild(o);                       //添加控件到窗体中
o = null;                                           //释放对象
}
</script>
</head>
<body onload="addInput();">
</body>
</html>
```

【运行效果】

动态创建按钮的运行效果如图 2-2 所示。

【难点剖析】

本例的重点是“createElement”方法，是 DOM 对象创建元素的方法，创建完元素后，指定元素的类型、值和方法，最后使用“appendChild”方法，将元素添加到 body 中。



图 2-2 动态创建按钮的运行效果

2.4 平面按钮

【实例描述】

默认按钮样式有点三维效果，有时为了与页面风格统一，需要将按钮设置为平面效果。本例介绍平面按钮的制作。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type=submit value=保存 style="border:1px ;solid:#666666; height:35px; width: 50px;
font-size:15pt;
background-color : #E8E8FF; color:#666666" name="submit">
<input id="Button1" type="button" value="保存" style="height:35px; width: 50px;
font-size:15pt" />
</body>
</html>
```

【运行效果】

平面按钮与普通按钮的对比效果如图 2-3 所示。

【难点剖析】

本例的难点是 CSS 样式表的控制。在按钮控件中，“style”属性用来设置控件的样式。“border”样式表示设置按钮的边框宽度，它是设置平面按钮的关键因素。



图 2-3 平面按钮与普通按钮的对比效果

2.5 按钮的嵌入效果

【实例描述】

Web 按钮可通过 CSS 实现不同的效果，本例介绍按钮的嵌入效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
```



```
<style type="text/css">
.over {color:yellow; background: navy}
.down {color:yellow; background: navy; font-style: italic}
</style>
</head>
<body>
<input
type="Button"
onMouseOver="this.className='over';"
onMouseOut="this.className='';"
onMouseDown="this.className='down';"
onMouseUp="this.className='over';"
value="让按钮嵌入"
onClick="this.value='嵌入成功! ' " name="Button">
</body>
</html>
```

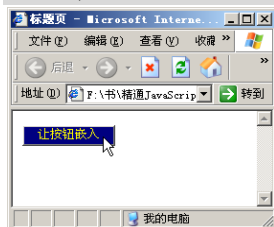


图 2-4 按钮的嵌入效果

【运行效果】

按钮的嵌入效果如图 2-4 所示。

【难点剖析】

本例的重点是 CSS 和按钮的鼠标事件。当鼠标移到按钮上时，触发的是“onMouseOver”事件，调用的样式是“over”；当鼠标按下时，触发的是“onMouseDown”事件，调用的样式是“down”。

2.6 改变状态栏信息按钮

【实例描述】

状态栏一般用来显示链接或网站的提示信息。本例通过单击按钮，实现状态栏内容的改变。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<form>
<input type="button" value="修改状态栏" onClick="self.status='欢迎光临我们的工作室!';"
name="button">
</form>
</body>
</html>
```

【难点剖析】

本例的难点在于获取窗体中的状态栏。“self”表示当前窗体，“status”表示窗体的状态栏。

此处也可以使用“window.status”获取窗体的状态栏。

2.7 定义按钮的热键

【实例描述】

如果界面内容比较多，为了方便用户操作，可为界面中的按钮提供热键。本例学习如何为按钮设置热键。

【实现代码】

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>使用热键</title>
</head>
<body>
<form action="http://www.google.com" method="get" name="form1">
<textarea rows=5 cols=50></textarea>
<br><input type="submit" accessKey="S" value="提交(Alt+s)">
</form>
</body>
</html>
```

【运行效果】

热键实例的运行效果如图 2-5 所示。

【难点剖析】

本例中重点是“input”按钮控件的“accessKey”属性，其用来设置控件的热键，通常内容是一些大写的字母。设置热键后，可使用“Alt+热键”的形式来激活操作。

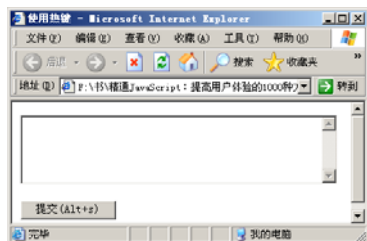


图 2-5 热键实例的运行效果

2.8 取得控件的绝对位置

【实例描述】

页面在布局时一般需要相对位置和绝对位置，本例学习如何获取控件的绝对位置。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function getAddress(e)
{
var t=e.offsetTop;
```

```
var l=e.offsetLeft;
while(e=e.offsetParent)
{
    t+=e.offsetTop;           //获取 x 坐标
    l+=e.offsetLeft;          //获取 y 坐标
}
alert("x 坐标="+t+"          y 坐标为="+l);
}
</script>
</head>
<body>
<input id="Button1" type="button" value="坐标" onclick="getAddress(this)" />
</body>
</html>
```

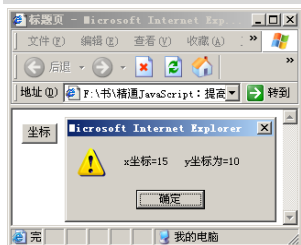


图 2-6 实例的运行效果

【运行效果】

实例的运行效果如图 2-6 所示。

【难点剖析】

本例的重点有两个：如何将当前元素作为参数传递；如何获取绝对的(x,y)坐标。传递当前控件使用“this”，获取绝对 x 坐标使用“offsetTop”属性，y 坐标使用“offsetLeft”属性。

2.9 删除时的确认提示

【实例描述】

如果用户不小心单击了“删除”按钮，则会损失一些重要的数据。为了让删除操作更安全，通常在用户单击“删除”按钮后，给出一个提示，让用户确认此次操作是否正确。这种提示就是本实例要介绍的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
    <script language="javascript">
        function del()
        {
            if(confirm("确实要删除吗?"))
                alert("已经删除!");
            else
                alert("已经取消了删除操作");
        }
    </script>
</head>
```

```
<body>
<input id="Button1" type="button" value="删除" onclick="del()" />
</body>
</html>
```

【运行效果】

删除时的提示界面如图 2-7 所示。

【难点剖析】

本例中使用了“confirm”方法，其提供一个小型对话框，允许用户继续或取消当前操作，如果方法返回“true”，则继续执行，否则终止当前操作。

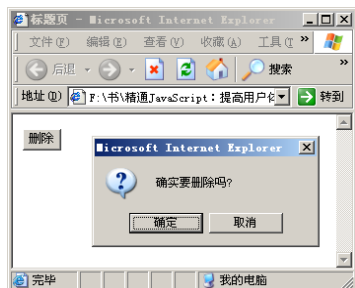


图 2-7 删除时的提示界面

2.10 按钮只能单击一次

【实例描述】

页面中的按钮通常是可以多次单击的，但有时候为了确定用户的选择，当按钮单击一次后，便不允许再使用。本例学习这种效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type="button" name="btn" value="单击" onclick="this.disabled=true">
单击上面的按钮测试是否只能按一次
</body>
</html>
```

【难点剖析】

本例的重点是按钮的“onclick”事件，以及控制按钮是否可用的属性“disabled”。当用户单击按钮后，便立即设置按钮的“disabled”属性为“true”，按钮会变为灰色，表示无法使用。代码中的“this”表示当前单击的按钮对象。

2.11 防止按钮连击

【实例描述】

当页面提交的数据特别多时，页面会反应比较迟钝，此时如果用户等不及而连续单击按钮，导致数据重复提交。可以使用本例提供的代码防止数据重复提交。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```



```
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function doubleCheck(){
    if (window.document.readyState != null && window.document.readyState != 'complete')
        //判断页面是否执行完毕

    {
        alert("正在处理, 请等待! ");           //没有执行完毕提示, 并返回 false
        return false;
    }
    else
    {
        return true;                           //执行完毕
    }
}
</script>
</head>
<body>
<input type=text name="txt1" value="this is test!">
<input type=button value="提交" onClick="doubleCheck()">
</body>
</html>
```

【难点剖析】

本例的重点是如何判断页面的状态。“readyState”属性用来获取页面的状态，其值只能获取，不允许赋予。当其值为“complete”时，表示页面已经加载完毕。

2.12 图片式按钮

【实例描述】

在很多网站开发工具中,都提供一个图片式按钮控件,但标准的 HTML 并不提供这个控件。本例学习如何通过变通的方法,自定义一个图片按钮。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language=JavaScript>
function goTo()
{

    var myindex=document.myform.mailBox.selectedIndex
    //获取下拉框中的选择索引
```

```

        location=document.myform.mailBox.options[myindex].value;
                                                //获取下拉框的选择值
    }
</script>
<form name="myform">
    <br />
    <select name="mailBox" size=1>
        <option selected>选项</option>
        <option value="http://www.163.net">163 电子邮箱</option>
        <option value="http://www.263.net">263 电子邮箱</option>
    </select><br />
    <a href="javascript: goTo()" onMouseOver="self.status='';return true" onMouseOut=
"self.status='';return true">
        </a>
    </form>
</body>
</html>

```

【运行效果】

图片式按钮的运行效果如图 2-8 所示。

【难点剖析】

本例首先通过下拉列表的“selectedIndex”属性，获取用户的选择。然后指定导航地址“location”。主要的图片放在一个“a”标签内，实现图片按钮的效果和功能。单击图片时，会实现页面的导航。

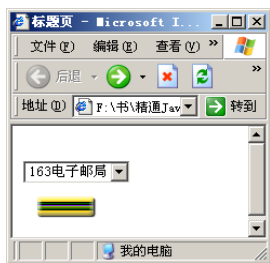


图 2-8 图片式按钮的运行效果

2.13 在按钮底部显示文字

【实例描述】

HTML 中的标准按钮，默认文字显示在按钮的中间，有时候为了界面的布局，需要设置按钮的文字。本例学习如何设置按钮的文字使其显示在底部。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<button style="width:100px; height:100px;padding-top:80px;">看看底部的文字</button>
</body>
</html>

```

【运行效果】

文字显示在按钮底部的效果如图 2-9 所示。



图 2-9 文字显示在按钮底部的效果

【难点剖析】

本例其实利用的是按钮控件的样式表。“padding-top”属性表示文本显示在按钮上的位置，此时一定要注意先设置按钮的高度，然后设置此属性。

2.14 选择不同的列表项时显示不同的按钮

【实例描述】

根据用户的选择不同，可显示不同的按钮，这需要学习动态创建按钮的技术。本例学习通过一个下拉列表的选择动态创建按钮，并设置按钮的单击事件。

【实现代码】

```
<HTML>
<HEAD>
<title>测试列表选择项 </title>
<script>
function butSelect(){
    var selVal = document.getElementById("sel").value;
    if(selVal == "1"){
        document.getElementById("td").innerHTML = '<input type="button" value="按钮 1"
onclick="btnc1();">';
    }else if(selVal == "2"){
        document.getElementById("td").innerHTML = '<input type="button" value="按钮 2"
onclick="btnc2();">';
    }else{
        document.getElementById("td").innerHTML = '';
    }
}
function btnc1(){
    alert("单击了按钮 1");
}

function btnc2(){
    alert("单击了按钮 2");
}
</script>
</HEAD>
<BODY>
<table>
<tr>
<td>
<select onChange="butSelect();" id="sel">
<option value="" >
```

```
<option value="1">but1
<option value="2">but2
</select>
</td>
<td id="td"></td>
</tr>
</table>
</BODY>
</HTML>
```

【运行效果】

本例的运行效果如图 2-10 所示。

【难点剖析】

本例的重点是如何将动态创建的按钮添加到指定位置，以及如何为动态按钮绑定单击事件。“innerHTML”是大部分容器控件都具备的属性，表示容器内的 HTML 代码，因为动态生成的“input”控件是一段 HTML 文本，所以不能使用类似“value”的属性将按钮添加到容器中。在创建按钮时，直接指定按钮的“click”事件与一个已有的方法关联。

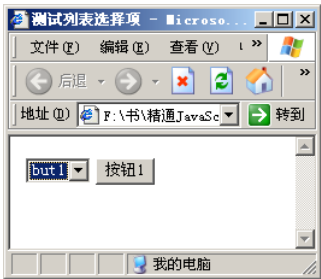


图 2-10 本例的运行效果

2.15 使用按钮控制文本渐变

【实例描述】

文本渐变是一种吸引用户眼球的特效，本例通过两个按钮，实现对文本渐变的控制。当用户单击“开始渐变”按钮时，文本就会实现渐变的效果，当单击“结束渐变”按钮时，文本还原成初始状态。

【实现代码】

```
<SCRIPT LANGUAGE="JavaScript">
var x=9;
var change="on"
if (navigator.appName == "Netscape") { //浏览器是 Netscape 的情况
    visShow='show'; //显示的命令
    visHide='hide'; //隐藏的命令
    docStyle="document.";
    styleDoc="";
}
else { //浏览器是 IE 的情况
    visHide="hidden"; //显示的命令
    visShow="visible"; //隐藏的命令
    docStyle="";
    styleDoc=".style"; //设置样式的命令
}
```



```
function hide1() {
eval(docStyle+ 'object1' + styleDoc + '.visibility=' + visHide);
//隐藏第一个 div
}
function hide2() {
eval(docStyle+ 'object2' + styleDoc + '.visibility=' + visHide);
//隐藏第二个 div
}
function hide3() {
eval(docStyle+ 'object3' + styleDoc + '.visibility=' + visHide);
//隐藏第三个 div
}
function hide4() {
eval(docStyle+ 'object4' + styleDoc + '.visibility=' + visHide);
//隐藏第四个 div
}
function hide5() {
eval(docStyle+ 'object5' + styleDoc + '.visibility=' + visHide);
//隐藏第五个 div
}
function hide6() {
eval(docStyle+ 'object6' + styleDoc + '.visibility=' + visHide);
//隐藏第六个 div
}
function hide7() {
eval(docStyle+ 'object7' + styleDoc + '.visibility=' + visHide);
//隐藏第七个 div
}
function hide8() {
eval(docStyle+ 'object8' + styleDoc + '.visibility=' + visHide);
//隐藏第八个 div
}
function hide9() {
eval(docStyle+ 'object9' + styleDoc + '.visibility=' + visHide);
//隐藏第九个 div
}
function hide10() {
eval(docStyle+ 'object10' + styleDoc + '.visibility=' + visHide);
//隐藏第十个 div
}
function show1() {
eval(docStyle+ 'object1' + styleDoc + '.visibility=' + visShow);
//显示第一个 div
hide2(),hide3();
}
function show2() {
eval(docStyle+ 'object2' + styleDoc + '.visibility=' + visShow);
//显示第二个 div
```



```

hide1(),hide3();
}
function show3() {
eval(docStyle+ 'object3' + styleDoc + '.visibility=' + visShow);
//显示第三个div

hide2(),hide4();
}
function show4() {
eval(docStyle+ 'object4' + styleDoc + '.visibility=' + visShow);
//显示第四个div

hide3(),hide5();
}
function show5() {
eval(docStyle+ 'object5' + styleDoc + '.visibility=' + visShow);
//显示第五个div

hide4(),hide6();
}
function show6() {
eval(docStyle+ 'object6' + styleDoc + '.visibility=' + visShow);
//显示第六个div

hide5(),hide7();
}
function show7() {
eval(docStyle+ 'object7' + styleDoc + '.visibility=' + visShow);
//显示第七个div

hide6(),hide8();
}
function show8() {
eval(docStyle+ 'object8' + styleDoc + '.visibility=' + visShow);
//显示第八个div

hide7(),hide9();
}
function show9() {
eval(docStyle+ 'object9' + styleDoc + '.visibility=' + visShow);
//显示第九个div

hide8(),hide10();
}
function show10() {
eval(docStyle+ 'object10' + styleDoc + '.visibility=' + visShow);
//显示第十个div

hide9();
}
function changel() {
x+=1;
eval("show" + x + "()");
//逐个显示div,从1到10
if (x<10) setTimeout("changel()", 75);
//一直在此方法中循环
else if (change=="on") change2();
//开始调用第二个循环方法
}

```

```
function change2() {  
    x-=1; //逐个显示 div, 从 10 到 1  
    eval("show" + x + "()");  
    if (x>1) setTimeout("change2()", 75); //一直在此方法中循环  
    else change1(); //开始调用第一个循环方法  
}  
function changeOn() { //开始变化的控制  
    x=9;  
    change="on";  
    change1();  
}  
function changeOff() { //结束变化的控制  
    change="off";  
}  
</SCRIPT>
```



图 2-11 文本渐变的效果

【运行效果】

文本渐变的效果如图 2-11 所示。

【难点剖析】

本例的重点是“change1”和“change2”方法。通过“change1”方法的循环执行，分别调用显示 div 的 10 个方法，然后再通过“change2”方法的循环，调用隐藏 div 的 10 个方法。代码中有个技巧“eval("show" + x + "()")”，使用“eval”方法可以将这些普通字符串，连接成一个函数名。

2.16 带翻页效果的公告栏

【实例描述】

公告栏通常需要 marquee 实现滚动效果，有时候因为页面的排版问题，可能需要翻页形式的公告栏。本例就学习如何实现这种功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<style type="text/css">  
#divMsg{  
    line-height:20px;  
    height:20px;  
    overflow:hidden;  
}  
</style>  
<script type="text/javascript">  
var Scroll = new function(){  
    this.delay = 2000; //延迟的时间
```

```

this.height = 20; //行的高度
this.step = 4; //步长
this.curHeight= 0;
this.stimer = null;
this.timer = null;
this.start = function(){ //开始翻页，调用 move 方法
    this.move();
}
this.move = function(){
    var self = this;
    if(this.curHeight == this.height) //如果显示完一行
    {
        this.timer = setTimeout(function() //使用定时器，定时下一行的翻页时间
            self.move();
        }, this.delay);
        this.curHeight = 0;
        if(this.element.scrollTop >= this.element.scrollHeight - this.height){
            //滚动信息已经完毕

            this.element.scrollTop = 0;
        }
        return true;
    }
    this.element.scrollTop += this.step;
    this.curHeight += this.step;
    this.timer = setTimeout(function(){ //设置自动翻页定时器
        self.move();
    }, 30);
}
this.stop = function(){ //清除定时期，停止滚动翻页
    clearTimeout(this.timer);
}
}
</script>
</head>
<body>
<div id="divMsg">
    张三奥运会历史性的突破，拿到了男子 100 米金牌<br/>
    奥运会历史上的首位 8 金得主<br/>
    北京奥运会欢迎志愿者的参与<br/>
    奥运会带来了什么样的商机<br/>
    北京奥运会 2008 年举行<br/>
    娱乐新闻请转到娱乐主页<br/>
    今天又获得一枚金牌<br/>
</div><script type="text/javascript">
Scroll.element = document.getElementById('divMsg');
Scroll.start();
</script>
<input type="button" value="开始" onclick="Scroll.start()"/>

```



```
<input type="button" value="停止" onclick="Scroll.stop()"/>
</body>
</html>
```



图 2-12 带翻页效果的公告栏

【运行效果】

带翻页效果的公告栏如图 2-12 所示。

【难点剖析】

本例的重点是显示信息高度的判断。代码中默认每行的高度为 20，如果当前行的高度已经超过 20，则调用定时器，根据指定的时间再显示下一行的信息。

2.17 动态设置控件的事件

【实例描述】

有时候需要在网页中动态添加一些控件，为了某些操作，又必须为这些控件添加事件，以达到与用户交互的目的。本例学习如何动态设置控件的事件。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
//添加事件的参数，参数为控件的标识
function addClick(obj)
{
    obj.onclick=function()                //绑定按钮的单击事件
    {
        alert('动态添加事件成功');        //单击事件完成的功能——输出提示
    }
}
</script>
</head>
<body>
    <input id="Button2" type="button" value=" 测试" />
    <br /><input id="Button1" type="button" value="为上面的按钮添加事件" onclick=
"addClick(Button2)" />
</body>
</html>
```

【难点剖析】

本例的重点是如何为控件绑定事件。绑定控件时需要知道此控件的唯一标识（ID 或 Name），然后以属性的方式指明要实现的事件（如 onclick）。事件的内容是一个方法。

第3章

字符串文本和输入框特效

本章导读

文本是网页开发中不可缺少的内容，用户通过这些文本了解网页的信息。字符串是文本的主要类型，而input和textarea则是用户输入文本的主要控件。本章从字符串、文本、控件三个方面入手，详细讲解网页中常见的文本操作技巧。

3.1 只带下画线的输入框

【实例描述】

在设计考试页面时，经常会碰到填空题，填空题的输入框要求只有一条下划线。本例学习此类输入框的设计。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
只有下画线的输入框<input type="text" name="txt1" size="25" style=" border:1px; border
-bottom-style: solid;border-top-style: none;border-left-style:none; border-right-style:
none; ">
<br />
默认的输入框<input id="Text1" size="25" type="text" />
</body>
</html>
```



图 3-1 两种输入框的对比
“border-bottom-style”。

【运行效果】

两种输入框的对比如图 3-1 所示。

【难点剖析】

本例的重点是边线样式的定义。如果要使控件有边框线，则需要将“border”样式设置为大于一的值，然后设置 4 个边框，左边框为“border-left-style”，右边框为“border-right-style”，上边框为“border-top-style”，下边框为

3.2 限定文本框可输入字符数

【实例描述】

在数据库中，数据字段的长度是固定的，为了阻止用户输入超长的数据，可通过自定义方法限定用户输入的数据长度。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function textcontrol(content)
{
```

```

        if (content.length > 2)           // 如果文本框的长度大于 200
            document.getElementById("txt").value=document.getElementById("txt"). value.
substring(0, 2);                          //截取前 200 字符
    }
</script>
</head>
<body>
<textarea id="txt" cols="50" rows="10" onPropertyChange="textcontrol(this.value)">
</textarea>
</body>
</html>

```

【难点剖析】

本例的重点有两个，即获取指定的文本框、截取特定长度的字符串。获取窗体内的文本框使用“document.getElementById("txt")”方法。截取长度是通过 String 对象的“substring”方法，其包含两个参数，起始地址和要截取的长度。

3.3 文字过长时的省略界面

【实例描述】

有时候为了在一个页面中显示所有的新闻信息，通常只显示内容的一部分，然后用省略号代替后面的内容。如果内容是固定的，则使用 HTML 元素可以解决；如果数据来自数据库，属于动态内容，该如何实现内容的省略呢？本例介绍如何动态实现文字过长时的省略界面。

【实现代码】

```

<script Language="javascript">
var sText="这是一段很长的文本，希望通过省略号实现";           //默认文本
var content;
function OmitText()
{
    content="<nobr>" +sText + "</nobr>";
    document.getElementById("mydiv").innerHTML=content;        //修改层的内容
}
</script>

```

在 body 中添加一个按钮和一个 id 为“mydiv”的层，代码如下所示：

```

<input type=button value=加载 onclick="OmitText()">
<DIV STYLE="width: 150px; height: 50px; border: 1px solid black;
        overflow: hidden; text-overflow:ellipsis" id="mydiv">
</DIV>

```

【运行效果】

文字过长时的省略界面如图 3-2 所示。

【难点剖析】

本例的重点是“<nobr>”标签，这是 DHTML 自带的标签，用来在不换行的情况下，实现

文本的修饰。本例中其主要功能是当 div 内的文本过长时，不对文本进行换行操作，而是以省略号替代超长的文本显示。

3.4 输出 26 个英文字母

【实例描述】

本例演示如何在浏览器中输出指定的字符。通过字符的 ASCII 值可以输出任意英文字母。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script>for(var i=65;i<91;i++)document.write(String.fromCharCode(i));</script></body>
</html>
```

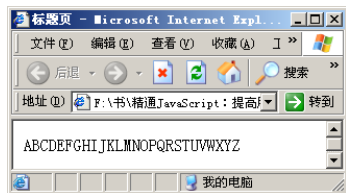


图 3-3 输出的 26 个英文字母

【运行效果】

输出的 26 个英文字母如图 3-3 所示。

【难点剖析】

本例的重点是英文字母的 ASCII 值和 String 对象的“fromCharCode”方法。大写英文字母的 ASCII 值从 65 开始，小写字母从 97 开始。“fromCharCode”方法用来将指定的 ASCII



图 3-2 文字过长时的省略界面

3.5 首字母变为大写

【实例描述】

在文本中出现英文时，为了符合标准的英文单词显示方式，有时需要将首字母改为大写。本例学习如何将单词设置为大写或小写。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function changeCase(frmObj)
{
var index;
var tmpStr;
var tmpChar;
```



```

var preString;
var postString;
var strLen;
tmpStr = frmObj.value.toLowerCase();           //转换为全部小写
strLen = tmpStr.length;                         //字符的长度
if (strLen > 0) {
for (index = 0; index < strLen; index++)
{
if (index == 0) {
//将第一位字符转换为大写
tmpChar = tmpStr.substring(0,1).toUpperCase();
postString = tmpStr.substring(1,strLen);
tmpStr = tmpChar + postString;
}
else {
tmpChar = tmpStr.substring(index, index+1);
//如果是第二个单词（通过空格判断）
if (tmpChar == " " && index < (strLen-1)) {
tmpChar = tmpStr.substring(index+1, index+2).toUpperCase();
preString = tmpStr.substring(0, index+1);
postString = tmpStr.substring(index+2,strLen);
tmpStr = preString + tmpChar + postString;
}
}
}
}
frmObj.value = tmpStr; //显示转换后的文本
}
</script>
</head>
<body>
<input type=text name="txt1" value="this is test!">
<input type=button value="转换文本" onClick="javascript:changeCase(txt1)">
</body>
</html>

```

【运行效果】

首字母变为大写的效果如图 3-4 所示。

【难点剖析】

本例的重点是判断首字母并变为大写。判断首字母通过单词与单词间的空格实现，而修改字母为大写，则使用的是 string 的方法“toUpperCase”。变为小写字母的方法为“toLowerCase”。

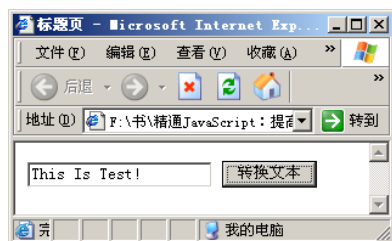


图 3-4 首字母变为大写的效果



3.6 textarea 自适应文字行数

【实例描述】

textarea 是 HTML 中的文本元素，可实现文字的多行输入，也可以控制行数和列数。本例学习如何让 textarea 根据用户的输入文本，自动调整高度和宽度。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<textarea rows=1 name=txt1 cols=27 onpropertychange="this.style.posHeight=this.
scrollHeight">
</textarea></body>
</html>
```

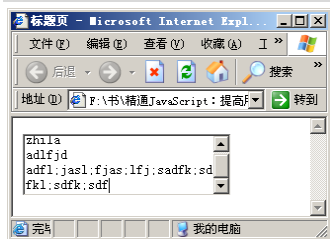


图 3-5 textarea 自适应文字行数的效果

【运行效果】

textarea 自适应文字行数的效果如图 3-5 所示。

【难点剖析】

本例的重点是 textarea 元素的“onpropertychange”事件。当文本内容发生变化时会触发此事件。“posHeight”表示文本输入框的高度，“scrollHeight”表示内容的高度。

3.7 禁止文本的复制和粘贴

【实例描述】

为了禁止信息的非法传播，很多网站禁止了文本复制和粘贴的功能，本例学习如何禁止复制和粘贴。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<textarea cols=50 rows=5 oncopy="document.selection.empty()" onpaste= "return false">
测试是否可以复制粘贴</textarea>
</body>
</html>
```

【运行效果】

实例的运行界面如图 3-6 所示。

本例的重点是“oncopy”和“onpaste”事件。“oncopy”表示用户的复制事件，当用户复制文本时，返回的是“document.selection.empty()”，表示不选择任何内容，即使用户选择了内容，此事件依然返回空。“onpaste”表示用户的粘贴事件，本例此事件返回“false”，表示不执行任何操作。

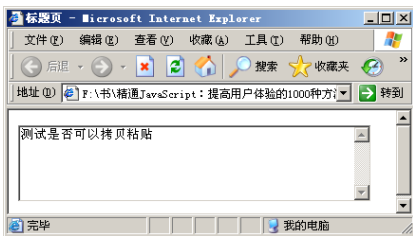


图 3-6 实例的运行界面【难点剖析】

3.8 控制两个文本框只输入其一

【实例描述】

在进行页面验证的时候，为了确保注册用户的唯一性，一般都通过邮箱或者姓名判断，但只能有一个判断的条件，即要么邮箱唯一，要么姓名唯一。本例学习如何控制两个文本框只有其中一个有值。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function checkFields()
{
    name = document.submitform.name.value;           //获取姓名
    email = document.submitform.email.value;           //获取邮箱
    if ((name == "") && (email == "")) {                //如果两者都为空
        alert("请输入姓名和邮箱!");
        return false;
    }
    else
        return true;
}
</script>
</head>
<body>
<form name=submitform onSubmit="return checkFields()">
    名字和邮箱只需要输入一个即可<br>
    <br>
    <table border=0><tr>
    <td align=center>名字</td><td> </td>
    <td align=center>Email</td></tr><tr>
    <td align=center><input type=text name=name value="" onFocus="document. submitform.
email.value='';" size=10></td>
    <td align=center>或</td>
    <td align=center><input type=text name=email value="" onFocus="document. submitform.
```



```
name.value='';" size=10></td>
</tr><tr>
<td colspan=3 align=center><input type=submit value="提交"></td>
</tr></table>
</form>
</body>
</html>
```

【难点剖析】

本例的重点在于文本框的“onfocus”事件，此事件在文本框获得焦点时被触发。当文本框获取焦点时，会将另一个文本框的值清空，实现代码为“document.submitform.email.value=""”。

3.9 判断编辑器中是否包含特殊字符

【实例描述】

因为网页中的标签都使用“<...>”来标注，如果用户在文本框中输入这些字符，可能会导致页面提交失败。本例提供一个简单的方法，用来检测用户的输入中是否包含特殊字符。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var bForbidden = false;
var ch;
var strForbidden = new Array("<",">","."); //罗列所有被禁止的方法字符
function chk(str)
{
    for (var i=0;i<strForbidden.length;i++){ //遍历用户输入的数据
        for (var j=0;j<str.length;j++)
        {
            ch=str.substr(j,1);
            if (ch==strForbidden[i]) //如果包含非法字符
            {
                bForbidden = true; //设置此变量为 true
            }
        }
    }
    if (bForbidden){
        alert("包含特殊字符,已经被禁止!");
    }
}
</SCRIPT>
</head>
<body>
<input type=text name="txt1" value="this is test!">
<input type=button value="测试文本" onClick="javascript:chk(txt1.value)">
```

```
</body>
</html>
```

【运行效果】

本例的运行效果如图 3-7 所示。

【难点剖析】

本例将所有的非法字符存放在一个数组“strForbidden”中，然后使用“for”语句遍历用户输入的字符，同时还要遍历非法字符所在的数组，这样一个字符一个字符地进行比较，从而严查用户输入的正确性。

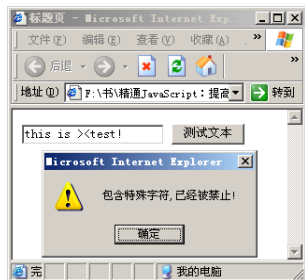


图 3-7 本例的运行效果

3.10 判断文本中回车的数量

【实例描述】

因为文本框允许自动换行，所以无法判断文本的实际行数，在将文本保存到文件中时，有时需要判断文本的行数，本例介绍如何通过回车判断行数。

【实现代码】

```
<HTML>
<HEAD>
<title>反选</title>
<script language=javascript>
function getCount()
{
    var count=document.all('TextAreal').innerHTML.split('\n').length-1;//获取回车的个数
    alert("总共有"+ count +"个回车");
}
</script>
</HEAD>
<BODY>
    <textarea id="TextAreal" cols="30" rows="8"></textarea>
    <input id="Button1" type="button" value="获取" onclick="getCount()" />
</BODY>
</HTML>
```

【难点剖析】

本例的重点是回车的表示符号“\n”。使用 string 对象的“split”方法，根据“\n”符号将文本切割成多个数组，通过数组的长度就可以判断回车的数量。

3.11 判断字符串中有多少汉字

【实例描述】

对于字符串的判断，使用最多的是正则表达式。本例学习如何利用正则，实现对汉字个数

的判断。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="JavaScript">
function cal(str)
{
    re=/[\u4E00-\u9FA5]/g;                //测试中文字符的正则
    if(re.test(str))                      //使用正则判断是否存在中文
        return str.match(re).length      //返回中文的个数
    else
        return 0
}
</script>
<input onblur="alert(cal(this.value))"></body>
</body>
</html>
```

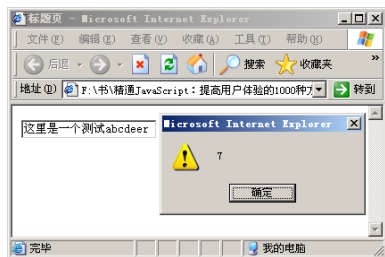


图 3-8 本例的运行效果

【运行效果】

本例的运行效果如图 3-8 所示。

【难点剖析】

本例的重点是正则表达式的应用步骤。

- (1) 创建一个正则表达式变量“re”；
- (2) 使用“re”的“test”方法，判断是否能检索出符合条件的字符串；
- (3) 使用“match”方法，计算符合条件的字符串的个数。

3.12 去除字符串前后的空格

【实例描述】

如果用户输入的内容中有空格（如输入姓名），通常在数据库中不容易显示，为了清楚地保存用户的数据，在保存信息时，通常需要去除信息前后的空格。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
function KillSpace(txtvalue)
```

```

{
    //判断文本内容是否为空，判断第一个字符是否是空格
    if((txtvalue.value.length>0) && (txtvalue.value.charAt(0)==' '))
        txtvalue.value = txtvalue.value.substring(1,txtvalue.value.length);
                                                //从第二位开始取

    //判断文本的长度
    if(txtvalue.value.length>0 && (txtvalue.value.charAt(txtvalue.value.length-1)==' '))
        txtvalue.value = txtvalue.value.substring(0,txtvalue.value.length-1);
                                                //去掉最后一位
}

</script>
</head>
<body>
<input type="text" name="txt1" value=" 空格 " />
<input type="button" value="去除空格" name="btn1" onclick="KillSpace(txt1)" />
</body>
</html>

```

【难点剖析】

本例的重点是字符串的截取操作。首先使用“charAt(0)”判断文本的第一位是否为空格，如果是，则使用“substring”方法，截取从第一位开始的字符串。然后同样的方法判断最后一位是否为空格，是则使用“substring”方法截取正确的字符串。

3.13 刷新时清空所有文本框

【实例描述】

用户刷新页面时需要清空页面中已经填写的内容，此时可以使用本例提供的方法，实现所有文本框的清空。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body onload="document.forms[0].reset()">
<form name="form1">
<input type="text" name="txt1" value="">
<input type="text" name="txt2" value="">
<input type="text" name="txt3" value="">
</form>
</body>
</html>

```

【难点剖析】

本例的重点是对“reset”方法的使用。HTML 提供了一个“input”类型的控件“reset”，用来清空页面文本框中的数据。本例内部工作原理就是调用的“reset”方法，此方法可以使页面

中的所有文本框重置（还原为初始状态）。如果文本框初始情况下是有值的，则无法实现清空。

3.14 随意改变大小的文本框

【实例描述】

网页中的文本框通常会固定大小，以保证页面的统一布局。本例学习如何动态改变文本框的大小。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<div contenteditable="true">
<input type='text'><button>改变这个本框试验试验</button>
</div></body>
</html>
```

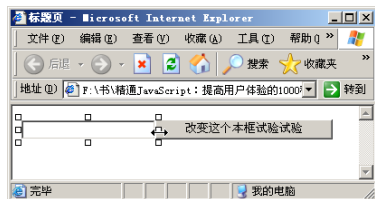


图 3-9 改变文本框大小时的效果

【运行效果】

改变文本框大小时的效果如图 3-9 所示。

【难点剖析】

本例的重点是 div 元素的“contenteditable”属性，此属性可将 div 的内容设置为可编辑。本例中将“input”控件放在 div 中，可实现“input”控件的动态改变。

3.15 文本框的自动全选

【实例描述】

本例提供便捷的用户操作，在一些开放源代码的网站，用户可以选择全部代码内容，并进行复制，如果代码比较多，可使用“全选”按钮。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function selectAll()
{
mytxt.select();
}
</script>
```



```

</head>
<body>
<textarea id="mytxt" cols=30 rows=6 >
</textarea>
<input id="mybtn" value="全选" type="button" onclick="selectAll()" />
</body>
</html>

```

【运行效果】

实例的运行效果如图 3-10 所示。

【难点剖析】

本例中使用了“select”方法，可以实现文本框内容的全部选择，此方法对所有的“input”元素都有效。

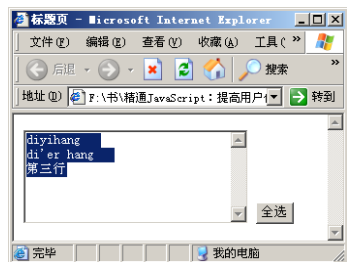


图 3-10 实例的运行效果

3.16 文本框滚动导航

【实例描述】

大部分的滚动导航都通过 div 层实现。本例将学习一个新的方法，使用文本框滚动导航，并能在文本框获取焦点时，实现窗口的自动导航。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<SCRIPT LANGUAGE="JavaScript">
function makeArray(q){
for(i=1 ; i < q ; i++){this[i]=0}}           //设置从 1 开始的数组
w=1;
howmanysites=4; // 显示导航的内容个数

Sites = new makeArray(howmanysites);         //设置导航数组

Sites[1] = "http://www.google.com|全球的搜索引擎!"; //以“|”间隔站点和描述
Sites[2] = "http://www.baidu.com|中国的搜索引擎!";
Sites[3] = "http://www.yahoo.com|还有一个搜索引擎!";
function showSites() {
    if (w > howmanysites) { w=1; };           //如果导航到结尾，则从头开始
    var string=Sites[w] + "";
    var split=string.indexOf("|");             //通过间隔符分组
    var url=string.substring(0,split);

```



```
var word=string.substring(split + 1,string.length); //获取链接的描述
document.form.url.value=url; //获取链接的 URL 地址
document.form.word.value=word;
w+=1;
window.setTimeout('showSites()',5000); //每隔 5 秒钟改变一下链接
}
function visitSite() {
    window.location=document.form.url.value; //导航到指定位置
}
</SCRIPT>
<center>
<form name=form>
<table><tr><td align=center>
<input type=hidden name=url value="">
<input type=text name=word value="" onFocus="visitSite()" size=40>
</td></tr></table>
</form>
</center>
<script>
showSites();
</script>
</body>
</html>
```

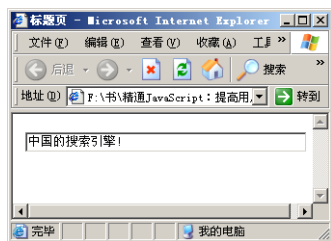


图 3-11 文本框滚动导航的效果

时，使用“window.location”实现窗口的导航。

【运行效果】

文本框滚动导航的效果如图 3-11 所示。

【难点剖析】

本例的重点是导航数组和文本框的“onFocus”事件。导航数组中的每项都包含两个值，即“URL 地址”和“URL 的描述信息”。这两个值以“|”间隔。通过字符串的“split”方法分割出这两个值，并将“URL 的描述信息”显示在文本框中。当文件框获得焦点时，使用“window.location”实现窗口的导航。

3.17 按钮获取焦点

【实例描述】

用户在注册时，需要阅读有关协议，并同意这些协议才允许继续注册。为了简化用户的操作，通常将焦点默认在“同意”按钮上。本例学习如何在页面打开时，让按钮自动获取焦点。

【实现代码】

```
<script language="javascript">
function GetFocus()
{
    document.form1.btnAgree.focus();
}
```

```
}
</script>
```

需要在页面中，添加一个 form 和两个按钮，注意按钮必须命名。在 body 的加载事件中调用“GetFocus”方法，代码如下所示：

```
<body onload="GetFocus()">
<form name="form1">
<input id="btnAgree" type="button" value="我同意"/>
<input id="btnRefuse" type="button" value="我不同意"/>
</form>
```

【运行效果】

默认焦点的界面如图 3-12 所示。

【难点剖析】

本例中使用了 form，要获取焦点的控件必须在 form 内，且焦点必须具备“name”或“id”属性，用来获取控件的唯一性标识。方法“focus”用来为指定的控件获取焦点。



图 3-12 默认焦点的界面

3.18 文本框获取焦点弹出下拉框

【实例描述】

为了避免用户输入错误，很多文本框（例如城市、国家等）不允许用户输入，而是弹出下拉框让用户直接选择。本例就学习如何让文本框获得焦点时，自动弹出下拉框供用户选择。

【实现代码】

```
<script LANGUAGE="JavaScript">
var oRegion = document.getElementById("txtRegion"); //需要弹出下拉列表的文本框
var oDivList = document.getElementById("divList"); //要弹出的下拉列表
var oClose = document.getElementById("tdClose"); //关闭div的单元格，也可使用按钮实现
var colOptions = document.getElementsByTagName("li"); //所有列表元素
var bNoAdjusted = true;
oClose.onclick = function()
{
    oDivList.style.display = "none"; //隐藏div，实现关闭下拉框的效果
};
//设置下列选择项的一些事件
for (var i=0; i<colOptions.length; i++)
{
    colOptions[i].style.cursor = "hand";
    colOptions[i].onclick = function() //为列表项添加单击事件
    {
        oRegion.value = this.innerText;
    };
};
```

```
colOptions[i].onmouseover = function()           //为列表项添加鼠标移动事件
{
    this.style.backgroundColor = "#ffff00";
};
colOptions[i].onmouseout = function()             //为列表项添加鼠标移走事件
{
    this.style.backgroundColor = "";
};
}
//文本获得焦点时的事件
oRegion.onfocus = function()
{
    oDivList.style.display = "block";
    if (bNoAdjusted)                               //控制 div 是否已经显示的变量
    {
        bNoAdjusted = false;
        //设置下拉列表的宽度和位置
        oDivList.style.width = this.offsetWidth + 50;
        oDivList.style.posTop = oDivList.offsetTop + this.offsetHeight;
        oDivList.style.posLeft = oDivList.offsetLeft - this.offsetWidth - 8;
    }
};
</script>
```



图 3-13 弹出下拉框的效果

【运行效果】

弹出下拉框的效果如图 3-13 所示。

【难点剖析】

本例的重点是如何获取用户的选择内容。弹出下拉框并不复杂，当文本框获取焦点时，设置 div（此处代表下拉框）的“style”属性为“none”即可。要获取用户选择的内容，则首先需要为窗体中所有的列表元素“li”设置一些事件，可参考代码中的详细注释。

3.19 文本框简单的单击效果

【实例描述】

为了提示用户文本框的输入内容，可以为文本框设置默认值，如“请输入姓名”。但如果这样，用户就需要先删除这个提示，然后再输入姓名。为了方便用户操作，本例实现单击文本框后，提示内容自动消失的特效。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```

<head>
<title>标题页</title>
</head>
<body>
<input name="username" type="text" onClick="this.value='' value="单击我试试" size="20"
maxlength="20">
</body>
</html>

```

【难点剖析】

本例中借用文本框控件的“onClick”事件，触发用户单击文本框的操作，“this.value”中“this”表示当前单击的对象，也就是输入框控件。“value”表示输入框的显示值。

3.20 文字的打字效果

【实例描述】

打字效果是将一段文本逐个文字地显示，实现打字的效果。本例学习如何制作这样的特效。

【实现代码】

```

<script language=javascript>
var layers =document.layers;
var style=document.all;
var both=layers||style;
var idme=908601;
if(layers)
{ layerRef='document.layers';styleRef ='';}
if(style)
{ layerRef='document.all';styleRef = '.style';}
//开始参数的定义
function writeOnText(obj,str)
{
if(layers)with(document[obj])
{ document.open();document.write(str);document.close();}
if(style)eval(obj+'.innerHTML=str');
}
var dispStr=new Array("证监会称将严查利用内幕信息牟取不当利益行为!");//要出现的文本
var overMe=0;
//逐字显示的方法
function txtTyper(str,idx,objId,objStyle,color1,color2,delay,plysnd)
{
var mystr='',strchar='';
var skip=200;
if (both && idx<=str.length) {
if (str.charAt(idx)=='<') { while(str.charAt(idx)!='>') idx++;}
if (str.charAt(idx)=='&'&&str.charAt(idx+1)!=' '){ while (str.charAt(idx)!=';')idx++;}
mystr = str.slice(0,idx); //返回数组从开始到指定位置的字符串

```



```
strchar = str.charAt(idx++); //当前地址的字符
if (overMe==0 && plysnd==1)
{
//针对浏览器的不同,调用不同的显示方法
if (navigator.plugins[0]){
if(navigator.plugins["LiveAudio"][0].type=="audio/basic" && navigator.javaEnabled())
{document.embeds[0].stop();
setTimeout("document.embeds[0].play(false)",100);}
} else if (document.all){
ding.Stop();
setTimeout("ding.Run()",100);}
overMe=1;}else overMe=0;
writeOnText(objId, "<span class="+objStyle+"><font color='"+color1+"'>"+ mystr+"
</font><font color='"+color2
+"'">"+strchar+"</font></span>");
setTimeout("txtTyper('"+str+"', "+idx+", '"+objId+"', '"+objStyle+"', '"+color1+"',
 '"+color2+"', "+delay+" ,"+plysnd+")",delay);}}
function init()
{txtTyper(dispsStr[0], 0, 'div1', 'style1', '#66CCBB', '#000000', 400, 0);}
</script>
```

需要在 body 中,添加一个 div,代码如下所示:

```
<BODY onload=init(>
<DIV class=style1 id=div1></DIV>
</BODY>
```

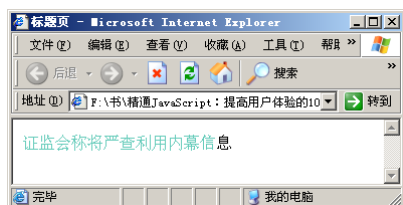


图 3-14 打字效果

【运行效果】

打字效果如图 3-14 所示。

【难点剖析】

本例的重点其实是实现这种效果的思路,代码并不是关键。在本例中,要实现一段文本的打字特效,首先将这段文本放在数组中,然后利用两个变量,实现打字和显示的效果,“打字”变量只是显示光标当前的某个字符,而“显示”变量则显示当前光标前所有的字符。

3.21 文字滚动

【实例描述】

很多网站的公告栏,由于内容太多,一般通过文字滚动形式显示。本例学习如何实现文字的滚动。

【实现代码】

```
<script language="javascript">
var oMarquee = document.getElementById("mydiv"); //滚动对象
```

```

var iLineHeight = 42;           //单行高度，像素
var iLineCount = 7;            //实际行数
var iScrollAmount = 1;         //每次滚动高度，像素
function play()
{
    oMarquee.scrollTop += iScrollAmount;
    if ( oMarquee.scrollTop == iLineCount * iLineHeight )
        oMarquee.scrollTop = 0;
    if ( oMarquee.scrollTop % iLineHeight == 0 ) {
        window.setTimeout( "play()", 2000 );
    } else {
        window.setTimeout( "play()", 50 );
    }
}
oMarquee.innerHTML += oMarquee.innerHTML;
window.setTimeout( "play()", 2000 );           //定时器用来循环滚动
</script>

```

需要在 body 中添加一个名为“mydiv”的层，详细代码参考随书光盘。

【运行效果】

文字滚动的效果如图 3-15 所示。

【难点剖析】

本例中，因为在 JavaScript 的方法外调用了“document.getElementById("mydiv")”，即调用 body 中的元素。因为脚本代码在运行时是逐行解释的，所以必须定义了 mydiv 元素，才可以使用上面的调用，因此此例中的代码段放在 div 的后面。文字滚动用此代码实现主要是为了控制方便，其实使用 HTML 的 marquee 元素也可以实现文字的滚动效果。

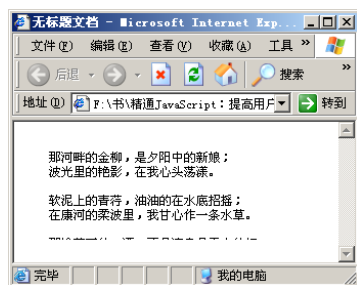


图 3-15 文字滚动的效果

3.22 文字滑动

【实例描述】

文字滑动和文字滚动的效果一样，本例提供一种自定义 marquee 类的方式，实现一段文字的滑动效果。

【实现代码】

```

<html>
<head>
<script type="text/javascript">
function $(id) {
    return document.getElementById(id);           //获取控件的 ID
}
function marquee(time, oDiv, oLtd, oRtd)           //制作一个 marquee 类
{

```



```

/*
time 值越大速度越慢
oDiv 最外层 div
oLtd 左边的 td
oRtd 右边的 td
*/
var timer, width = oLtd.offsetWidth;
function move() {
    if (oDiv.scrollLeft >= width)                //当滚动条移动到最后时, 重新开始
        oDiv.scrollLeft = 0;
    else
        oDiv.scrollLeft ++;                    //一直滚动
}

oRtd.innerHTML = oLtd.innerHTML;                //左侧内容转移动到右侧
oDiv.style.overflow = "hidden";                //隐藏最外层 div
oRtd.style.width = oLtd.offsetWidth;            //右侧 td 的宽度
oDiv.onmouseover = function () {                //鼠标移动过来的事件
    window.clearInterval(timer);                //清空定时器-停止滚动
};
oDiv.onmouseout = function () {
    timer = window.setInterval(move, time);        //鼠标移走便开始滚动
};
timer = window.setInterval(move, time);        //开始滚动
};

window.onload = function () {
    new marquee(25, $("myDiv"), $("myTd1"), $("myTd2"));
}
</script>
</head>
<body>
<div id="myDiv" style="border:#CCCCCC 1px dashed; width:300px;"><!--设置显示的宽度-->
    <table cellspacing="0" cellpadding="0">
        <tbody>
            <tr>
                <td id="myTd1">
                    <table width="342" cellpadding="0" cellspacing="0"><!--注意这里的宽度必须设置, 并且要设置为具体值-->
                        <tr align="center">
                            <td>左边</td>
                            <td>右边</td>
                            <td>左边</td>
                            <td>右边</td>
                        </tr>
                    </table>
                </td>
                <td id="myTd2"></td>
            </tr>
        </tbody>
    </table>

```



```

        </tr>
    </tbody>
</table>
</div>
</body>
</html>

```

【运行效果】

文字的滑动效果如图 3-16 所示。

【难点剖析】

本例的重点是自定义的 `marquee` 类，其中定义了 `marquee` 的“onmouseover”、“onmouseout”事件和“move”方法。通过此类可轻松实现文字或图片的滑动效果。



图 3-16 文字的滑动效果

3.23 文字跳动特效

【实例描述】

本例的文字跳动特效主要是为了增添网页的特色，实现类似于用凸透镜浏览文字的效果。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
function nextSize(i,incMethod,textLength)           //获取随机字体大小的方法
{
    if (incMethod == 1) return (72*Math.abs( Math.sin(i/(textLength/3.14)))) );
    if (incMethod == 2) return (255*Math.abs( Math.cos(i/(textLength/3.14)))));
}
function sizeCycle(text,method,dis)
{
    output = "";
    for (i = 0; i < text.length; i++)
    {
        size = parseInt(nextSize(i +dis,method,text.length));
                                                //获取随机的字体大小
        output += "<font style='font-size: "+ size +"pt'">" +text.substring (i,i+1)+ "</font>";
    }
    myDiv.innerHTML = output;                    //动态输出指定字体大小的文本
}
function doWave(n)
{
    theText = "欢迎光临";                        //要浏览的文本
    sizeCycle(theText,1,n);                       //实现跳动的方法
    if (n > theText.length) {n=0}                //如果文本显示完毕,则从头再开始
}

```



```
        setTimeout("doWave(" + (n+1) + ")", 50);    //不断调用该方法，实现循环作用
    }
</script>
</head>
<body bgcolor="#fef4d9" onload=doWave(0);>
<div ID="myDiv" align="center">
</div></body>
</html>
```

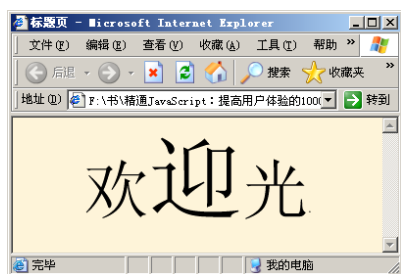


图 3-17 文字跳动特效

【运行效果】

文字跳动特效如图 3-17 所示。

【难点剖析】

本例的重点其实是动态改变文字的字体大小。本例通过“nextSize”方法获取随机的字体大小，然后设置文本中每个字的大小不同，通过“innerHTML”属性，将文本动态显示在 div 中，从而实现跳动的效果。

3.24 荧光效果的文本

【实例描述】

本例的主要目的是增添网页的视觉效果，学习制作具有荧光效果的文本。

【实现代码】

```
script language="JavaScript">
function SymError()
{
    return true;
}
window.onerror = SymError;
var from = 1;
var to = 4;
var delay = 55;                                // 闪的速度
var glowColor = "#FFCC00";                     // 颜色
var i = to;
var j = 0;
textPulseDown();
// 向上跳动的方法
function textPulseUp()
{
    if (!document.all)
        return
    if (i < to)
    {
        theText.style.filter = "Glow(Color=" + glowColor + ", Strength=" + i + ")";
```

```

i++;
setTimeout('textPulseUp()',delay);
return 0;
}
if (i = to)
{
setTimeout('textPulseDown()',delay);
return 0;
}
}
//向下跳动的方法
function textPulseDown()
{
    if (!document.all)
        return
    if (i > from)
    {
        theText.style.filter = "Glow(Color=" + glowColor + ", Strength=" + i + ")";
        //设置文本的滤镜效果

        i--;
        setTimeout('textPulseDown()',delay);
        //设置定时器
        return 0;
    }
    if (i = from)
    {
        setTimeout('textPulseUp()',delay);
        return 0;
    }
}
</script>

```

需要在 body 中添加一个 ID 为 “theText” 的文本，可参考随书光盘。

【运行效果】

荧光效果的文本如图 3-18 所示。

【难点剖析】

本例的重点是 JavaScript 对象的 “filter” 滤镜样式，其对应了 CSS 中的 “Glow” 滤镜。对一个对象使用 “Glow” 属性后，这个对象的边缘就会产生类似发光的效果。



图 3-18 荧光效果的文本

3.25 文字逐个闪亮——霓虹灯效果

【实例描述】

常见的霓虹灯效果，是通过文字逐个循环显示实现。本例通过文本的颜色变换，实现文字的霓虹灯效果。



【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
text = "欢迎光临我们的俱乐部";           //显示的文字
color1 = "gray";                          //文字的颜色
color2 = "blue";                          //转换的颜色
fontsize = "6";                           //字体大小
speed = 100;                              //转换速度（毫秒）
i = 0;
if (navigator.appName == "Netscape") {    //浏览器不同，输出的标签不同
    document.write("<layer id=myDiv visibility=show></layer><br><br><br>");
}
else {
    document.write("<div id=myDiv></div>");
}
function changeCharColor()
{
    if (navigator.appName == "Netscape") { //Netscape 浏览器的情况下
        document.myDiv.document.write("<center><font face=arial size =" + fontsize +
"><font color=" + color1 + ">");
        for (var j = 0; j < text.length; j++) {
            if(j == i) {
                document.myDiv.document.write("<font face=arial color=" + color2 + ">" +
Text.charAt(i) + "</font>");
            }
            else {
                document.myDiv.document.write(text.charAt(j));
            }
        }
        document.myDiv.document.write('</font></font></center>');
        document.myDiv.document.close();
    }
    if (navigator.appName == "Microsoft Internet Explorer")
        //IE 浏览器的情况下
    {
        str = "<center><font face=arial size=" + fontsize + "><font color=" + color1 + ">";
        for (var j = 0; j < text.length; j++) { //循环输出指定字体大小和颜色的文本
            if( j == i) {
                str += "<font face=arial color=" + color2 + ">" + text.charAt(i) + "</font>";
            }
            else {
                str += text.charAt(j);
            }
        }
        str += "</font></font></center>";
    }
}
```

```

        myDiv.innerHTML = str;                                //在div中显示文本
    }
    (i == text.length) ? i=0 : i++;                            //如果i的值不大于文本的长度，则自增
}
setInterval("changeCharColor()", speed);                    //通过定时器，实现不断的循环
</script>
</head>
<body>
</body>
</html>

```

【运行效果】

文字的霓虹灯效果如图 3-19 所示。

【难点剖析】

本例的重点是霓虹灯效果的原理，霓虹灯就是逐个点亮一段文字。本例中通过循环获取当前要点亮的字符，然后根据指定的颜色和字体大小，设置此字符的样式，本例中用颜色变化代表点亮某个字符。修改后的文本再用 div 动态输出，以实现文本的动态改变效果。

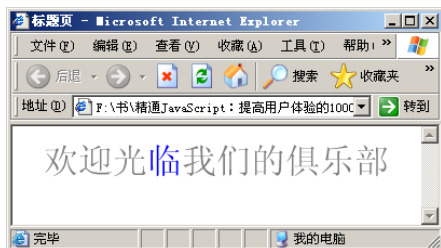


图 3-19 文字的霓虹灯效果

3.26 旋转式的变色文字特效

【实例描述】

文本不仅可以实现垂直、水平滚动，而且还可以旋转式地滚动，本例通过 JavaScript 和 CSS 滤镜相结合的方式，实现一段旋转式的变色文本。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT language=javascript>
myWord="欢迎来看北京奥运会"                                //要旋转的文字
tmpStr=""                                                    //连接字符串
Taille=40;
wordLength=myWord.length;                                    //获取文本的长度
for (x=0;x<wordLength;x++)
{
    //动态添加层，设置层的 ID，和每个层上显示的文本
    tmpStr=tmpStr + '<DIV Id=L' + x + ' STYLE="width:3;font-family: Courier New;font-weight:bold;position:absolute;top:40;left:50;z-index:0">' + myWord.charAt(x) + '</DIV>'
}
document.write (tmpStr);                                      //输出动态 div 层，显示旋转文本
Time=window.setInterval("txtRound()",10);                  //使用定时器，实现文本的不断旋转
Alpha=5;

```



```
I_Alpha=0.05;
function txtRound()
{
    Alpha=Alpha-I_Alpha;                                //类似步长的功能
    for (x=0;x<wordLength;x++){
        Alpha1=Alpha+0.5*x;
        Cosine=Math.cos(Alpha1);
        Ob=document.all("L"+x);                        //获取指定的层
        Ob.style.posLeft=100+100*Math.sin(Alpha1)+50;    //设置 div 的 x 坐标
        Ob.style.zIndex=20*Cosine;                      //设置层的层叠顺序
        Ob.style.fontSize=Taille+25*Cosine;             //设置层的字体
        Ob.style.color="rgb(" + (27+Cosine*80+50) + "," + (127+Cosine*80+50) + ",0)";
                                                    //实现颜色的变化
    }
}
</SCRIPT>
</head>
<body>
</body>
</html>
```

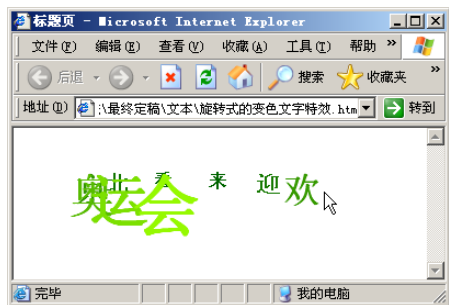


图 3-20 旋转式的变色文字特效效果。

【运行效果】

旋转式的变色文字特效如图 3-20 所示。

【难点剖析】

本例的重点是“Math.cos”方法。此方法以弧度为单位，计算并返回指定角度的余弦值。本例在旋转时，可以看到文字具有一定的弧度。每个文字都在一个 div 层上，通过“Math.cos”方法获取一个动态弧度，然后使用“zIndex”属性设置 div 的层叠顺序，最终实现旋转效果。

3.27 《黑客帝国》中的字符下落效果

【实例描述】

所有看过电影《黑客帝国》的人可能都对一个场景非常熟悉，那就是一连串的 0、1 编码。本例学习如何制作那种字符下落效果。

【实现代码】

```
<HTML>
<HEAD>
    <TITLE>黑客帝国</TITLE>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</HEAD>
```

```

<style type="text/css">
body
{
    overflow:hidden;
    margin:0;
    background-color:#000000;
    font-family:宋体;
}
DIV.#heike
{
    overflow:hidden;
    position:relative;
    top:5%;
    width:90%;
    height:90%;
    border-style:solid;
    border-width:1;
    border-color:#009900;
}
</style>
<script language="javascript">
var strCount;
var str;
var Color;
var Font;
var sLine = "W<br>W<br>W<br>.<br>B<br>a<br>i<br>D<br>u<br>.<br>C<br>O<br>M";
function OnLoad()
{
    strCount = 40;
    str = [];
    Color = [];
    Font = [];
    Color[0] = "#002211"; //文字的颜色
    Color[1] = "#003311";
    Color[2] = "#005511";
    Color[3] = "#008811";
    Color[4] = "#00BB99";
    Color[5] = "#114411";
    Color[6] = "#335566";
    Color[7] = "#668899";
    Color[8] = "#99BBAA";
    Color[9] = "#CECECC";
    Font[0] = "10px"; //文字的大小
    Font[1] = "12px";
    Font[2] = "14px";
    Font[3] = "16px";
    Font[4] = "18px";
    setTimeout("strik()",50);
}

```



```

}
function strik()
{
    for(var i=0;i<strCount;i++)
    {
        if(typeof(str[i]) != "undefined")                //如果字符串存在
        {
            if(str[i]["Carch"].style.pixelTop > heike.clientHeight)
            {
                str[i]["Carch"].outerHTML = "";
                delete str[i]["Level"];                    //删除数组元素
                delete str[i]["Speed"];
                delete str[i]["Carch"];
                delete str[i];
            }
            else
            {
                str[i]["Carch"].style.pixelTop += str[i]["Speed"];
            }
        }
        else if(Math.random()<0.25)                        //随机小数
        {
            str[i] = new Array();
            str[i]["Level"] = Math.round(Math.random()*4);
            str[i]["Speed"] = (Math.round(Math.random()*str[i]["Level"])) <<2)+10;
            document.all["heike"].insertAdjacentHTML("AfterBegin", "<span
id='SPAN_"+i+"'>"+sLine+"</span>");
            str[i]["Carch"] = document.all["SPAN_"+i];
            str[i]["Carch"].style.fontSize = Font[str[i]["Level"]];
                                                                    //字体
            str[i]["Carch"].style.position = "absolute";          //位置
            str[i]["Carch"].style.pixelLeft = Math.round(Math.random(
*heike.clientWidth);                                            //x坐标
            str[i]["Carch"].style.pixelTop = -str[i]["Carch"].offsetHeight;
                                                                    //y坐标
            str[i]["Carch"].style.color = Color[str[i]["Level"]]+5;
                                                                    //颜色
            str[i]["Carch"].style.filter = "glow(Color="+Color[str[i]
["Level"]]+",Strength=5)";                                       //滤镜效果
            str[i]["Carch"].style.zIndex = str[i]["Level"];       //z-Index
        }
    }
    setTimeout("strik()",50);
}
</script>
<BODY onload="OnLoad()">
<table width="100%" height="100%" border="0" cellspacing="0" cellpadding="0">
<tr><td align="center" height="100%"><div id="heike"></div></td></tr>

```



```

<tr><td align="center" style="padding:5;font-size:9pt;color:#FFFFFF;">使用 IE 6.0 以上版本或以 IE 为核心的浏览器浏览本页, 1024*768 分辨率为佳</td></tr>
</table>
</BODY>
</HTML>

```

【运行效果】

单击单元格后的效果如图 3-21 所示。

【难点剖析】

本例的重点是对颜色和速度的随机设置。JavaScript 中的“Math”对象用来提供数学运算,其中“Math.random”用来获取一个 0 到 1 之间的随机数。“Math.round”是采用四舍五入方式取得最接近的整数。代码中使用了二维数组,“delete”方法用来删除数组中的元素。

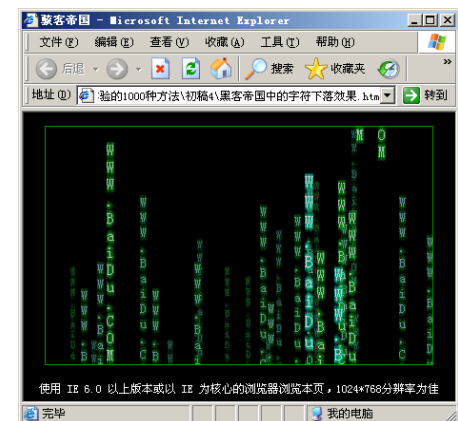


图 3-21 单击单元格后的效果

3.28 获取表单中文本框的个数

【实例描述】

因为 HTML 中的很多控件都是“input”元素,所以必须使用类型来判断其中某种控件的个数。本例演示如何获取表单中文本框的个数。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function checkform()
{
    var icount=0;
    var frm=document.getElementById("form1").getElementsByTagName("input");
    //获取窗体中所有的 input 元素

    for(var i=0;i<frm.length;i++){
    //遍历每一个 input 元素
    if(frm[i].type=="text"){
    //判断 input 元素的类型
    icount++;
    //是则给统计变量+1
    }
    }
    alert("总共有"+icount+"个文本框");
}
</script>
</head>
<body>
<form action="" name="dd" method="get" id="form1">
<input name="" type="text">
<input name="" type="text">

```



```
<input name="" type="text">
<input name="提交" value="提交" type="submit" onclick="checkform()">
</form>
</body>
</html>
```

【难点剖析】

本例的重点有两个，即获取页面中所有的 input 元素，通过类型判断哪些是文本框。获取所有 input 元素使用的是“getElementsByTagName”方法，判断 input 类型使用的是“type”属性。

3.29 光标停在文本框最后

【实例描述】

当用户修改文本框的内容时，为了方便操作，可以将光标直接放在文本框的最后，则用户可直接输入。本例学习如何将光标放在文本框最后。

【实现代码】

```
<script language="javascript">
function last()
{
    var e = event.srcElement;
    var r = e.createTextRange();
    r.moveStart("character", e.value.length);
    r.collapse(true);
    r.select();
}
</script>
```

需要在 body 中，添加一个多行文本框，当其获取焦点时调用上面的方法，代码如下所示：

```
<textarea rows=6 cols=30 onfocus="last()">
这是一个测试；
这是一个测试；
这是测试；
</textarea>
```



图 3-22 文本框获取焦点后的效果

【运行效果】

文本框获取焦点后的效果如图 3-22 所示。

【难点剖析】

本例中使用了三个主要方法：createTextRange、moveStart 和 collapse。createTextRange 用来获取文本框当前光标的位置。moveStart 方法用来移动光标，包括两个参数：移动间隔类型和移动多少个间隔，本例的间隔类型是字符“character”。collapse 方法

只有一个布尔型的参数，其值为“true”时折叠到选定区域的开始处，“false”时正好相反。

3.30 分行取 textarea 中的值

【实例描述】

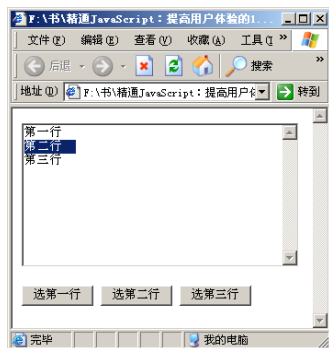
在文本域内选择文本时，可以一行一行地选择，同时这个选择是自动的。本例学习如何实现这种效果。

【实现代码】

```
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=gb2312">
<SCRIPT language="javascript">
function getRange(num, areaId)                                //行号，文本区域的 ID
{
    var txtRange = document.all(areaId).createTextRange();    //获取鼠标
    var rect = txtRange.getClientRects();                      //选择范围
    var left = rect[0].left;                                    //左侧位置
    if(num > rect.length - 1)                                  //超出行范围
        return;
    if(num == 0)                                                //如果是第一行
    {
        var right = rect[0].right;
        txtRange.moveEnd("character",-txtRange.text.length);    //移动到结尾
        while(txtRange.offsetLeft + txtRange.boundingWidth < right) //没有到结尾
        {
            txtRange.expand("character");                        //扩展宽度到结尾
        }
        return txtRange;                                        //返回选择区域
    }
    else
    {
        var right = rect[num].right;                            //右侧范围
        var txtRange = getRange(num - 1, areaId);              //获取选择域
        txtRange.moveStart("character",txtRange.text.length + 1); //移动到开始位置
        while(txtRange.offsetLeft + txtRange.boundingWidth < right) //如果没有到结尾
        {
            txtRange.expand("character");                        //扩展到结尾
        }
        if(txtRange.offsetLeft > left)
            txtRange.moveStart("character",-1);                //开始位置前
        return txtRange;
    }
}
function getText(num)                                          //根据行号，返回选择
{
```



```
var txtRange = getRange(num,"mytxt")
if(txtRange != null)
{
    txtRange.select();           //如果不为空，则选择
}
}
</SCRIPT>
</HEAD>
<BODY>
<TEXTAREA cols="40" rows="10" id="mytxt">
第一行
第二行
第三行
</TEXTAREA><p>
<input type="button" onClick="getText(0)" value="选第一行">
<input type="button" onClick="getText(1)" value="选第二行">
<input type="button" onClick="getText(2)" value="选第三行">
</BODY>
</HTML>
```



【运行效果】

分行取 textarea 中值的效果如图 3-23 所示。

【难点剖析】

本例的重点是如何实现选择一段文本。“createTextRange”用来创建选择范围，“moveStart”、“moveEnd”、“expand”方法实现复杂选区，这些方法都接收两个参数：移动的单位 and 移动单位的个数。其中移动单位是固定值：“character”，“word”，“sentence”或“textedit”。

图 3-23 分行取 textarea 中值的效果

3.31 自动插入文本

【实例描述】

用户将光标放到文本中时，会自动将一段文本添加到光标处，这就是本例要实现的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
document.onclick =function(){           //重写 onclick 事件
    var sel = document.selection;       //判断文本的选择
    if (sel!=null) {
```

```

var rng = sel.createRange(); //获取光标位置（也可以是选择的一段文本）
if (rng!=null)
    rng.pasteHTML("<font color=blue>插入的文字</font>"); //插入 HTML 文本
}
}
</SCRIPT>
</head>
<body>
这是一段测试文本
</body>
</html>

```

【运行效果】

文本插入后的效果如图 3-24 所示。

【难点剖析】

本例的重点是光标位置的设置和文本的插入。“document.selection”表示当前网页中的选中内容，“createRange”会根据当前文字选择返回 TextRange 对象。“pasteHTML”用来插入带标签的文本。

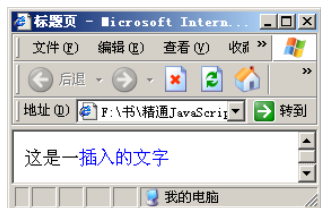


图 3-24 文本插入后的效果

3.32 选取 textarea 中的指定行

【实例描述】

使用控件的 focus 或 select 方法，可以很方便地选取 textarea 的所有内容，但却无法实现某行的选取。本例讲解如何选取 textarea 中的指定行。

【实现代码】

```

function getTxtRow(num, mytxt) //获取指定行的方法，第二个参数为文本框 ID
{
    //获取文本框内当前光标的位置
    var range = document.getElementById(mytxt).createTextRange();
    var rect = range.getClientRects(); //返回一个矩形
    var left = rect[0].left;
    if(num > rect.length - 1 || num < 0)
        return;
    if(num == 0) //选择第一行的情况
    {
        //设置选择范围
        var right = rect[0].right;
        range.moveEnd("character", -range.text.length);
        while(range.offsetLeft + range.boundingWidth < right)
        {
            range.expand("character");
        }
    }
}

```



```
        return range;
    }
    else
    {
        //设置选择范围
        var right = rect[num].right;
        var range = getTxtRow(num - 1, mytxt);
        range.moveStart("character", range.text.length + 1);
        while((range.offsetLeft + range.boundingWidth) < right)
        {
            range.expand("character");
        }
        if(range.offsetLeft > left)
            range.moveStart("character", -1);
        return range;
    }
}
//选择指定行数的方法
function getText(num)
{
    var range = getTxtRow(num, "txt")    //调用真正的获取行方法
    if(range != null)                    //如果指定的行内容不为空
    {
        alert(range.text);
        range.select();                  //选择指定的行
    }
}
</SCRIPT>
```

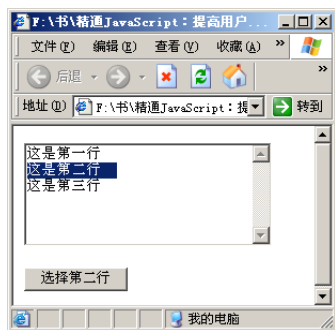


图 3-25 选中行的效果

【运行效果】

选中行的效果如图 3-25 所示。

【难点剖析】

本例的难点在于光标的获取，以及文本的选择范围。使用“createTextRange”方法获取文本中光标的位置。使用“getClientRects”选择一个范围，然后设置范围的起始位置和结束位置。最后使用“select”方法选择整行内容。

3.33 文本放大镜

【实例描述】

有些网页为了显示足够多的内容，把文本字体变得很小。为了让读者可以轻松地阅读文本，本例将介绍一种简单的文本放大镜。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<p>这是旧文本，一般大小</p>
<P onmouseover="this.style.zoom='180%'" onmouseout="this.style.zoom='normal'">
这是新文本，放大 180 倍
</p>
</body>
</html>

```

【运行效果】

文本的原始效果与放大效果的对比如图 3-26 所示。

【难点剖析】

本例的重点是对鼠标事件的调用,和 zoom 样式的使用。当用户将鼠标移动到文本上时,需要将文本放大,所以需要调用鼠标的“onmouseover”事件,如果鼠标离开时,则还原文本的大小,这需要调用鼠标的“onmouseout”事件。放大效果使用的是“zoom”,其使用百分比来设置放大的比例。

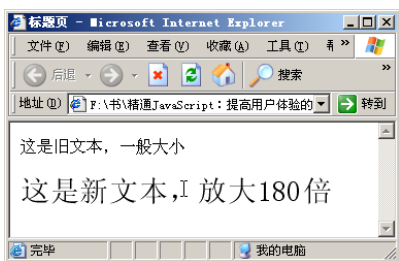


图 3-26 文本的原始效果与放大效果的对比

3.34 文本框的默认输入法

【实例描述】

网站的最终目的,是为用户提供方便,本例将通过输入法的自动切换,来提高网页录入的方便性。当用户要在英文文本框中输入内容时,可以自动切换系统的输入法为英文,如果在中文输入框中输入内容,可切换输入法为中文。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
默认: <input><br>
中文: <input style="ime-mode:active"><br>
英文: <input style="ime-mode:disabled"></body>
</html>

```

【难点剖析】

本例的重点是文本框的样式。“ime-mode”用来设置输入法,其主要包括一下 4 个属性:



active 代表输入法为中文；inactive 代表输入法为英文；auto 代表打开输入法；disable 代表关闭输入法。

3.35 文本框中显示网页中选中的内容

【实例描述】

为了方便用户的复制，可以将用户选择的内容显示在文本框内。本例学习如何在用户选择的同时，显示其选择的内容。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var txt1 = "";
function getSelect() {
    txt1 = (document.all) ? document.selection.createRange().text : document.
getSelection();                                // 获取当前选中的文本
    document.form1.txt1.value = txt1;          // 显示选择内容
    return true;
}
document.onmouseup = getSelect;                // 绑定鼠标事件
if (!document.all)
    document.captureEvents(Event.MOUSEUP);     // Netscape 下的鼠标事件捕获
</script>
</head>
<body>
<form name=form1>
<strong> 被选中的文本: <input type=text name=txt1 value=""></strong>
<div>
泛型
该语言中添加了一些泛型类型，使得程序员能够实现程度很高的代码重用，获得更高的集合类性能。泛型类型只存在
arity 上的不同。也可以将参数强制为特定的类型。有关更多信息，请参见泛型类型参数。
<br />
迭代器
迭代器使得规定 foreach 循环将如何循环访问集合的内容变得更加容易。
<br />
分部类
分部类型定义允许将单个类型（比如某个类）拆分为多个文件。Visual Studio 设计器使用此功能将它生成的代码
与用户代码分离。
</div>
</form>
</body>
</html>
```


【运行效果】

选择的同时显示选择内容的效果如图 3-27 所示。

【难点剖析】

本例的重点是如何获取文档中选定的文本，代码中使用了“document.selection.createRange().text”方法。“createRange”方法用于创建 TextRange 对象，其 text 属性用来获取用户选中的文本。



图 3-27 选择的同时显示选择内容的效果

3.36 文字的垂直滚动

【实例描述】

HTML 中的 marquee 元素可以实现文字的垂直滚动，但无法实现滚动属性的自定义。本例通过一段 JavaScript 代码，学习如何实现一段文本的垂直滚动。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
done = 0; //判断是否滚动
step = 4; //滚动的步长
function anim(yp,yk) //实现垂直滚动的方法
{
    if(document.layers) document.layers["divtxt"].top=yp; //Netscape 浏览器下
    else document.all["divtxt"].style.top=yp; //指定文本的 y 坐标
    if(yp>yk) step = -4 //如果已经到底部，则开始向上移动
    if(yp<60) step = 4 //如果已经到顶部，则开始向下移动
    setTimeout('anim('+(yp+step)+','+'+yk+')', 35); //循环执行滚动方法
}
function start()
{
    if(done) return
    done = 1;
    if(navigator.appName=="Netscape") {
        document.divtxt.left=innerWidth/2 - 145;
        anim(60,innerHeight - 60)
    }
    else
    {
        divtxt.style.left=11; //设置文本的 x 坐标
        anim(60,document.body.offsetHeight - 60) //调用垂直滚动的方法
    }
}
```



```
</script>
<div id="divtxt" style="position: absolute;top: -50;color: #000000;font- family:宋
体;font-size:9pt;">
<p><font color=blue>文字垂直滚动的特效演示</font>
</p></div>
<script language="JavaScript">
    setTimeout('start()',10);           //循环执行 start 方法
</script>

</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是实现垂直滚动的方法“anim”。其包含两个参数：“yp”和“yk”。“yp”参数表示顶端位置，到达此位置后文本不能再往上滚动。“yk”表示底端位置，到达此位置后，不能再往下滚动。其中文本滚动的原理，就是不断地动态改变文本的 y 坐标。

3.37 文字幻灯片

【实例描述】

幻灯片在图像切换中的应用非常广泛，有时为了增加页面的美观性，也需要为文字内容设置幻灯片。本例通过 JavaScript 和 CSS 的结合，学习制作文字幻灯片。

【实现代码】

```
<script language="JavaScript">
//用数组存放循环显示的信息
Text = new Array(
    "<a href='http://www.google.com' target='_blank' class='cr4'>我们的理工科学生必须要学
习语文</a>",
    "<a href='http://www.google.com' class='cr4'>英语不过不能毕业，现在汉语不过也不能毕业
</a>",
    "<a href='http://www.google.com' target='_blank' class='cr4'>要求所有理工科学生必须要
学习语言、文学等方面的两门...</a>",
    "<a href='http://www.google.com' target='_blank' class='cr4'>文化素质教育核心课程”北
京航空航天</a>",
    "<a href='http://www.google.com' target='_blank' class='cr4'>有必要。中国人国语不行还
谈什么</a>"
)
var IDX= -1;
//用来循环显示内容的方法
function playAd()
{
    if (IDX==Text.length-1) {
```

```

        IDX=0;
    } else {
        IDX++;
    }
    var prefix = "";
    divText.filters[0].apply();           //应用滤镜效果
    divText.innerHTML = prefix + Text[IDX]; //注意 divText 是表格中单元格的 ID
    divText.filters[0].play();
    to = setTimeout("playAd()",6000);     //定时器用来循环显示
}
</script>

```

需要在 body 中添加一个 table，用来定义内容显示的区域，代码如下所示：

```

<table width=453 border=0 cellpadding=0>
    <tr bgcolor=CCF4B9>
        <td height=30 bgcolor=CCF4B9 id=divText class=trans></td>
    </tr>
    <script>playAd()</script>
</table>

```

【运行效果】

文字幻灯片运行效果如图 3-28 所示。

【难点剖析】

本例的重点是对象滤镜的使用。JavaScript 为对象提供了“filters”属性，专门用于设置滤镜效果。其总共有三种方法，语法和注释如下所示：

```

对象名.filters(滤镜数值).Apply()    //装备滤镜
对象名.filters(滤镜数值).Play()      //开始播放滤镜效果
对象名.filters(滤镜数值).Stop()      //停止滤镜效果

```

3.38 随机动态文字效果

【实例描述】

随机性的动态文字，对文字的颜色、大小等都没有固定。此效果就是把一些不规则的字体拼凑在一起。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script>
    var s=new Array("LOVE","晚上","看","电影","9点","必须","秘密开始行动","如见","不到","请等待",
    "10点","不见","赶紧闪人!")
    function txtFont(str){

```



图 3-28 文字幻灯片运行效果



```
var n=Math.floor(Math.random()*5);
var obj1=document.createElement("B");           //创建黑体标签
var obj2=document.createElement("font");         //创建字体标签
obj2.size=Math.floor(Math.random()*5+4);         //随机字体大小
obj2.innerHTML=str
if(n>4){
    obj2.size=20                                //设置字体大小
    obj1.appendChild(obj2);                    //添加字体标签到黑体标签内
    return obj1.outerHTML                      //输出
}else{
    return obj2.outerHTML
}
}
for(var i=0;i<10;i++)
document.write(txtFont(s[i])+" ")              //输出内容
</script>
</head>
<body>
</body>
</html>
```

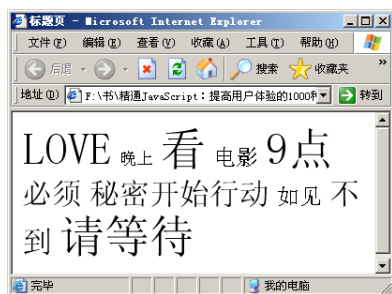


图 3-29 本例的运行效果

【运行效果】

本例的运行效果如图 3-29 所示。

【难点剖析】

本例的重点是随机字体的选取,以及动态创建标签。“B” 标签表示字体为“黑体”,“font” 标签用来设置字体的大小。“Math.random” 用来获取 0~1 之间的随机数,“Math.floor” 取数据的整数部分。

3.39 实现 textarea 的自动滚动

【实例描述】

很多页面的滚动效果都使用 marquee 来实现,有些时候也可以使用 scroll 属性实现。本例将利用 scroll 实现 textarea 的自动滚动。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body >
<button onclick="text1.scrollTop=text1.scrollHeight">滚动</button><br>
<textarea id="text1" cols=35 rows=6>
```

```

    第一行
    大锅饭
    adfasdf
    uityityui
    阿道夫
    afdasf
    e35qwfgesrreq
    adfsafsdf
    fasd
    lkjhljkg
    tet
    wrtwet
    最后一行
</textarea>
</body>
</html>

```

【运行效果】

滚动后的效果如图 3-30 所示。

【难点剖析】

本例的重点是一些 scroll 属性，scroll 是滚动条的意思。“text1.scrollTop”表示当前文本区域滚动条的 y 坐标。“text1.scrollHeight”表示滚动条的滚动区域（高度）。设置“text1.scrollTop=text1.scrollHeight”，表示将滚动条移到最底端。如果觉得本例中移动速度太快，可以通过重设定定时器解决速度问题。

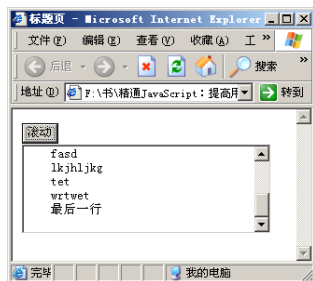


图 3-30 滚动后的效果

3.40 使用 marquee 实现文字上下滚动

【实例描述】

文字滚动可以通过定时器实现，也可以使用 HTML 中的 marquee 元素实现，本例介绍后者的使用情况。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>

<marquee direction=up scrollamount=1 scrolldelay=100 onmouseover='this. stop()'
onmouseout='this.start()'
height=60>
<table>
<tr>

```



```
<td>
这是第一行第一列
</td>
</tr>
<tr>
<td>
这是第二行第一列
</td>
</tr>
<tr>
<td>
这是第三行第一列
</td></tr>
</table>
</marquee>
</body>
</html>
```

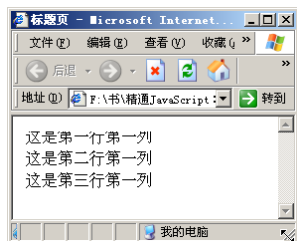


图 3-31 文字上下滚动的效果

【运行效果】

文字上下滚动的效果如图 3-31 所示。

【难点剖析】

本例的难点是对 marquee 元素的控制。滚动控件通常具备两个功能：鼠标经过停止和鼠标移走继续。这里通过 marquee 元素的事件“onmouseover”和“onmouseout”完成这两个功能。“stop”和“start”方法用来控制 marquee 是否滚动。

3.41 类似安装效果的 textarea 滚动

【实例描述】

现在使用 Microsoft 提供的 Install 工具，都是向导形式的安装，但一些比较底层的工具安装，通常会使用类似 DOS 中的 BAT 批处理效果。本例学习如何通过 textarea 的滚动，实现这种效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<textarea id="install" cols=20 rows=10 style="background-color:Black; color:White;
"></textarea>
<script>
var tmpStr = "";
for(var i = 0;i< 26;i++)
{
```

```

        for(var j=0;j<=i;j++)                //遍历 26 个英文字母
        {
            tmpStr += String.fromCharCode(65+i);    //连接要显示的字符串
        }
        tmpStr += ".....\r\n";                //行结束符号
    }
    var arr = tmpStr.split("\r\n");              //将字符串切割成数组
    function addLine(line)
    {
        if(line >= arr.length)
            return;
        var txtOb = document.getElementById("install");    //获取 textarea 控件
        txtOb.value += arr[line] + "\r\n";                //每局输出后都要求换行
        if(txtOb.clientHeight <= txtOb.scrollHeight)
            txtOb.scrollTop = txtOb.scrollHeight - txtOb.clientHeight;
        setTimeout("addLine(" + (line+1) + ")",500);
    }
    setTimeout("addLine(0)",1);
</script>
</body>
</html>

```

【运行效果】

本例的运行效果如图 3-32 所示。

【难点剖析】

本例的重点有两个：循环输出 26 个英文字母，以及逐行输出的设计原理。输出 26 个英文字母，是通过字符串对象的“fromCharCode”方法，其中“65”是大写字母“A”的 Unicode 编码。逐行输出使用的是换行符“\r\n”。



图 3-32 本例的运行效果

3.42 始终显示在最顶端的文本

【实例描述】

在很网站上可以看到一些广告总是随着页面的滚动而滚动，同时又能一直显示在页面的最顶端，其中的原理就是使用定时器不断改变 div 的位置。

【实现代码】

```

<script language="JavaScript">
    function StaticDiv(theName,Top,Left)
    {
        RealTop=parseInt(document.body.scrollTop);    //获取当前屏幕的 X 坐标
        TrueTop=Top+RealTop;
        document.all[theName].style.top=TrueTop;        //指定 div 的 X 坐标位置
        RealLeft=parseInt(document.body.scrollLeft);    //获取当前屏幕的 Y 坐标
    }

```

```
TrueLeft=Left+RealLeft;  
document.all[theName].style.left=TrueLeft;           //指定 div 的 Y 坐标位置  
}  
setInterval('StaticDiv("AlwaysDiv",0,0)',1);         //定时重置 div 的位置  
</script>
```

在页面中要设计始终显示的内容如下所示：

```
<div id="AlwaysDiv" style="position:absolute; left:1.5px; top:0px; width: 230px;  
height:30px; z-index:10">  
    <font size="5"><b>保持最顶端的文字</b></font>  
</div>
```

【运行效果】

始终在最顶端显示的文本实例的运行效果如图 3-33 所示。



图 3-33 始终在最顶端显示的文本实例的运行效果

【难点剖析】

本例中的要点有层和坐标。“scrollTop”是 body 的属性，用来获取最顶端的坐标，而“scrollLeft”用来获取最左侧的坐标。其中层用来显示文本内容，并使用了“setInterval”不断检测页面的左上角的坐标，以及时改变层的位置。

3.43 JavaScript 过滤 SQL 注入字符

【实例描述】

由于大多数的数据提交语句都是通过多个字符串进行连接，所以为了防止 SQL 的注入字符，本例介绍一种过滤特殊字符的方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
<script LANGUAGE="JavaScript">  
function check(inputStr) {  
    if (typeof(inputStr) != "string") { return inputStr; } //判断是否是字符串类型  
    var tmpValue = inputStr;
```



```
//以下搜索字符串中的特殊字符，如果存在，则替换成"
while (tmpValue.indexOf(';') > -1) {tmpValue = tmpValue.replace(';', ''); }
while (tmpValue.indexOf('<') > -1) {tmpValue = tmpValue.replace('<', ''); }
while (tmpValue.indexOf('>') > -1) {tmpValue = tmpValue.replace('>', ''); }
while (tmpValue.indexOf('--') > -1) {tmpValue = tmpValue.replace('--', ''); }
while (tmpValue.indexOf(",") > -1) {tmpValue = tmpValue.replace(",", ""); }
while (tmpValue.indexOf("'") > -1) {tmpValue = tmpValue.replace("'", ""); }
while (tmpValue.indexOf("?") > -1) {tmpValue = tmpValue.replace("?", ""); }
document.getElementById("txt1").value = tmpValue;      //重新显示更改后的变量
}
</script>
</head>
<body>
<input type=text id="txt1" value="select * from userinfo where username=zhang' and
passwrod=2" style="width: 392px">
<input type=button value="提交" onClick="check(txt1.value)">
</body>
</html>
```

【难点剖析】

本例的重点是对特殊字符的判断。SQL 需要防范的注入字符在代码中被一一列举，使用“indexOf”方法，可以判断字符串中是否包含这些字符。如果“indexOf”返回“-1”，则表示不包含指定的特殊字符。

3.44 textarea 内实现行的翻页效果

【实例描述】

textarea 中内容太多时，会出现滚动条，以支持对内容的浏览。本例学习如何通过两个按钮，实现行的上下滚动。

【实现代码】

```
<html>
<head>
<title>表格显示数据表记录</title>
</head>
<body>
<textarea id=myTxt style="overflow:hidden" width="20" height="20">
测试数据
测试数据
测试数据
测试数据
测试数据
测试数据
测试数据
测试数据
```

```
测试数据
测试数据
</textarea>
<button onmouseup=stop() onmousedown=up(>)上翻</button><button onmousedown= down()
onmouseup=stop(>)下翻</button>

<script>
var Timer=null;
function up()
{myTxt.doScroll("scrollBardown") //调用滚动条，实现上翻
  Timer=setTimeout("up()",100)
}
function down()
{myTxt.doScroll("scrollBarUp") //调用滚动条，实现下翻
  Timer=setTimeout("down()",100)
}
function stop()
{clearTimeout(Timer)} //清空定时器
</script>
</body>
</html>
```

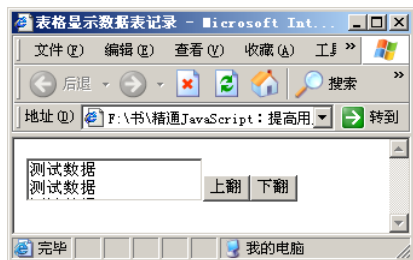


图 3-34 本例的运行效果

【运行效果】

本例的运行效果如图 3-34 所示。

【难点剖析】

本例中使用了 textarea 中的“doScroll”方法，以实现滚动条的手动拖动效果。其中“scrollBardown”表示将滚动条向上翻，“scrollBarUp”表示将滚动条向下翻。

3.45 textarea 中的文本插入

【实例描述】

在很多 HTML 编辑器中，可以轻松实现文本的增、删、改功能。本例将学习其中的插入或替换功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
//选择文本时保存光标位置-单击时同样
function storePos (txtobj)
{
  if (txtobj.createTextRange)
```

//获取选中的内容

```

        txtobj.caretPos = document.selection.createRange().duplicate();
    }
    function insertTextArea (txtobj, text)
    {
        if (txtobj.createTextRange() && txtobj.caretPos) {
            var caretPos = txtobj.caretPos;           //获取光标所在的位置
            //替换光标处位置
            caretPos.text =caretPos.text.charAt(caretPos.text.length - 1) ==' ' ?text + ' ' : text;
        }
        else
            txtobj.value = text;                       //直接显示插入的文本
    }
</script>
</head>
<body>
    <TEXTAREA NAME="mytxt" ROWS="5" COLS="25" WRAP="soft" onselect="storePos(this);"
onclick="storePos(this);" onkeyup="storePos(this);"> 实现文本的插入，文本框可以实现增、删、改功能
</TEXTAREA>
    <br />
    <INPUT TYPE="text" NAME="insertTxt" SIZE="20" VALUE="要插入的文本"><br />
    <INPUT TYPE="button" VALUE="插入文本" onclick="insertTextArea(mytxt, insertTxt.value);">
</body>
</html>

```

【难点剖析】

本例的重点是如何获取光标所在的位置。如果用户选择了一段文本，则使用“document.selection.createRange().duplicate()”获取光标位置。

3.46 查找两段文本中相同的词句

【实例描述】

在网站的常用操作中，有时需要对比两段文本内容，找出其中的相同点和不同点。本例学习如何找出两段文本内容的相同点。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
    <SCRIPT LANGUAGE="JavaScript">
function compare(a, b, n)
{
    var c=a.length>b.length?b:a;
    //减少循环
    if(b==c) b=a; a=c;
    if(!n) n=1;

```



```
//创建数组，逐字比较文本
var mm = new Array();
for(var i=0; i<a.length; i++)
{
for(var j=i+n; j<=a.length; j++)
{
var s = a.substring(i, j);
if(b.indexOf(s)==-1)
{
if(s.length>n) {
mm[mm.length] = a.substring(i, j-1);
i = j-2; }
break;
}
else {
if(j==a.length) {
mm[mm.length]=s; }
}
} }
return mm.Unique().join(","); //去除数组里的重复项
}
//为 Array 数组定义方法 Unique
Array.prototype.Unique = function()
{
var a = {};
for(var i=0; i<this.length; i++)
{
if(typeof a[this[i]] == "undefined") //判断文本是否有特殊类型
a[this[i]] = 1;
}
this.length = 0;
for(var i in a)
this[this.length] = i;
return this;
};
//定义变量，并调用比较方法
var a = "北京要举办奥运会";
var b = "上海要举办世博会";
alert("返回的结果: "+ compare(a, b, 2));
</SCRIPT>
</head>
<body>
</body>
</html>
```

【运行效果】

比较结果如图 3-35 所示。

【难点剖析】

本例的重点是使用两个循环逐字节比较字符，但这个并没有技术难点，最难的是“prototype”属性，用其提供对象的一组基本功能，本例中为 Array 对象创建了方法“Unique”。

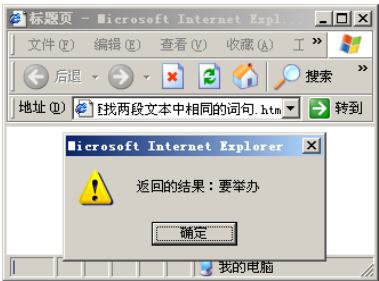


图 3-35 比较结果

3.47 自动保存网页的文本

【实例描述】

为了保存网页的一些内容，需要调用另存为对话框，实现内容的保存。本例演示如何快速保存网页的内容。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script>
window.onload=function()
{
    document.open("text/html"); //打开窗口
    document.write("把这些文字保存起来!!!!"); //输出内容
    document.execCommand("saveAs","true","我的网页"); //打开“另存为”对话框
    document.close(); //关闭文档
}
</script>
</head>
<body>
</body>
</html>
```

【运行效果】

自动保存网页内容的效果如图 3-36 所示。



图 3-36 自动保存网页内容的效果



【难点剖析】

本例的重点是“execCommand”方法。此方法用来执行浏览器中的一些常用操作，本例中使用“saveAs”调用“另存为”对话框，将网页的内容进行保存。

3.48 文本编辑器

【实例描述】

在网站提供的邮箱中，经常会使用文本编辑器，以实现文本内容的层次感和美观性。本例将制作一个简单的文本编辑器，主要学习实现文本编辑器的原理。

【实现代码】

```
<script language="javascript">
var edit;           //当前选择的文本编辑区域对象
var RangeType;     //对象类别
function start() //开始初始化编辑器——编辑区域是 Iframe
{
    Editor.document.designMode="ON";
    Editor.document.open();
    Editor.document.write(myTextArea.value);
    Editor.document.close();
    fnInit()
}
function setFocus() {
    Editor.focus(); //编辑器或去焦点
}
function selectRange(){
    edit = Editor.document.selection.createRange(); //编辑器的文本选择区域
    RangeType = Editor.document.selection.type;
}
//包装文本选定区域的执行命令
function execCommand(command,para) {
    setFocus();
    selectRange();
    if (para=="") //没有参数的情况
        edit.execCommand(command)
    else
        edit.execCommand(command, false, arguments[1]);
    Editor.focus();
    if (RangeType != "Control") edit.select();
}
//获取或设置文本的格式——字体、字号
function doSelectC(str,el) {

    var Index = el.selectedIndex;
    if (Index != 0)
```

```

        {   el.selectedIndex = 0;
            execCommand(str,el.options[Index].text);
        }
    }
    //获取或设置当前选定块的格式化标签
    function doSelectCl(str,el)
    {
        var Index = el.selectedIndex;
        if (Index != 0)
        {   el.selectedIndex = 0;
            execCommand(str,"<"+el.options[Index].value+">");
        }
    }
    //初始化
    function fnInit(){
        for (i=0; i<document.all.length; i++)
            document.all(i).unselectable = "off";           //指定不选中任何元素
        getSystemFonts();
    }
    //获取系统字体的方法
    function getSystemFonts()
    {
        var a=dlgHelper.fonts.count;
        var fArray = new Array();
        var oOption = document.createElement("OPTION");
        oOption.text = "字体";
        oOption.value = "0";
        selectFontName.add(oOption);
        //使用 DOM 方法 createElement 将字体依次添加到复选列表中
        for (i = 1;i < dlgHelper.fonts.count;i++)
        {
            fArray[i] = dlgHelper.fonts(i);
            var oOption = document.createElement("OPTION");
            oOption.text = fArray[i];
            oOption.Value = i;
            selectFontName.add(oOption);
        }
    }
    //格式化, 保全 script、textarea、xmp、pre 和 style 内容
    function formatfor(va) {
        var t=va.replace(/\r/g,'');
        t = t.replace(/(<(script|textarea|xmp|pre|style).*?>)([^\r]*?)(<\/\2>)/img,
        function () {return arguments[1]+arguments[3].replace(/\n/g, "\r")+ arguments[4]} )
        t = t.replace(/\n/g, "");
        return t
    }
    }
    function fontsize(el) //改变字体的方法
    {var Index=el.selectedIndex

```



```

var addpre="<font size="+el.options[Index].value+">"
if(Index>7)addpre="<font style='font-size:"+el.options[Index].value+"pt'>"
var oSel = Editor.document.selection.createRange()
var sBookmark = oSel.getBookmark()
var oSelhtml=oSel.htmlText
if(oSelhtml!="")
{
//定位选中内容
var conts=oSelhtml
var textLength = Editor.document.body.innerText.length
oSel.moveStart("character", -1*textLength)
var contp=formatfor(oSel.htmlText)
var conta=formatfor(Editor.document.body.innerHTML)
var contpa=''
var partC=""
var partB=""
var partA=""
var m=0
m=conta.indexOf(contp.substr(0,3))
var f=contp.length
for(;f>0;f--)
{if(conta.substr(m,f)==contp.substr(0,f)){contpa=contp.substr(0,f);partC=
conta.substr(m+f);break}}
var ko=contp.substr(f)
var kol=ko.length
var ty=conta.substr(m+f,kol)
var hu=""
for(var b=1;b<kol;b++)
if(ko.substr(b)==ty.substr(0,kol-b)){hu=ko.substr(b);
contpa+=hu;partC=partC.substr(kol-b);break}
var k=contpa.length
cont=conts.replace(/\n/g, "")
var u=cont.length
if(cont==contpa.substr(k-u)){partB=cont;partA=contpa.substr(0,k-u)}else{
for(u=cont.length;u>0;u--)
{if(cont.lastIndexOf(contpa.substr(k-u))!=-1){partB0=contpa.substr(k-u);partA0=
contpa.substr(0,k-u);break}}
contt=formatfor(conts)
if(hu!="")if(contt.substr(contt.length-kol)==ko)contt=contt.substr(0,contt.length-kol
)+hu
u=contt.length
var youm=contpa.lastIndexOf(contt)
if(youm!=-1){partB=contt;partA=contpa.substr(0,youm);partC=contpa.substr(youm+u)+
partC}else{
for(;u>0;u--){if(contt.lastIndexOf(contpa.substr(k-u))!=-1){partB1=contpa.substr(k-
u);partA1=contpa.substr(0,k-u);break}}
if(partB1.length>partB0.length){partB=partB1;partA=partA1}else{partB=partB0;partA=partA0}
}
}
}

```



```

    }
    if(partB.substr(partB.length-1)=="<"){partB=partB.substr(0,partB.length-1);partC="<"+
partC}
    if(partB.substr(partB.length-2)=="</"){partB=partB.substr(0,partB.length-2);partC="</
"+partC}
    //保护 textarea、 xmp、 script 和 style 的内容不被改变
    var cook=[]
    partA=partA.replace(/(<(script|textarea|xmp|style).*?>)[\s\S]*?</\2>/ig,
    function () {co=cook.length
    cook[co]=arguments[0];return "<cook"+co+">"})
    var ook=""
    partA=partA.replace(/(<(script|textarea|xmp|style).*?>)[\s\S]*?$/i,
    function () {co=cook.length
    ook=arguments[2]
    cook[co]=arguments[0];return "<cook"+co+">"})
    if(ook!=""){fd="(^[\s\S]*?</"+ook+">)"
    jk=new RegExp(fd,["i"])
    if(jk.test(partB)){jk.exec(partB)
    co=cook.length
    cook[co]=RegExp.$1
    partB=partB.replace(jk,"<cook"+co+">")}}
    partB=partB.replace(/(<(script|textarea|xmp|style).*?>)[\s\S]*?</\2>/ig,
    function () {co=cook.length
    cook[co]=arguments[0];return "<cook"+co+">"})
    ook=""
    partB=partB.replace(/(<(script|textarea|xmp|style).*?>)[\s\S]*?$/i,
    function () {co=cook.length
    ook=arguments[2]
    cook[co]=arguments[0];return "<cook"+co+">"})
    if(ook!=""){fd="(^[\s\S]*?</"+ook+">)"
    jk=new RegExp(fd,["i"])
    if(jk.test(partC)){jk.exec(partC)
    co=cook.length
    cook[co]=RegExp.$1
    partC=partC.replace(jk,"<cook"+co+">")}}
    partC=partC.replace(/(<(script|textarea|xmp|style).*?>)[\s\S]*?</\2>/ig,
    function () {co=cook.length
    cook[co]=arguments[0];return "<cook"+co+">"})

    //处理字体
    var dol=[]
    var dos=[]
    var lon=[]
    fd="<FONT([ ^>]*?>)"
    jk=new RegExp(fd,["im"])
    while(jk.test(partB)){ce=dol.length
    jk.exec(partB)
    dol[ce]=RegExp.$1

```



```

partB=partB.replace(jk,"<site"+ce+">")
}
ce=dol.length-1
for(;ce>-1;ce--)
{kjc="<site"+ce+">"
fd=kjc+'(.*)</font>'
jk=new RegExp(fd,["im"])
if(jk.test(partB)){dos[dos.length]=ce
jk.exec(partB)
ko3=kjc+RegExp.$1+"</site"+ce+">"
partB=partB.replace(jk,ko3)
}else{lon[lon.length]=ce}
}
partB=partB.replace(/</font>/img,"<lonf>")
for(var c=dos.length-1;c>-1;c--)
{
uts=dol[dos[c]]
if("==(uts.replace(/size=[0-7+]/i,"").replace(/style=(\"|')FONT-SIZE:
[0-9.]+.*;*(\"|')/im,"").replace(/[\s\n\r]/mg,"")){partB=partB.replace("<site"+dos[c
]+>","")
partB=partB.replace("</site"+dos[c]+>","")}else{partB=partB.replace("<site"+dos[c]+
>","<font"+uts.replace(/ size=[0-7+]/im,"").replace(/FONT-SIZE:
[0-9.]+pt/im,"").replace(/ style=(\"|');*(\"|')/im,"")+>")
partB=partB.replace("</site"+dos[c]+>","</font>")}
}
//处理其他标签
var addend=""
var mio=[]
partB=partB.replace(/<([^<> ]*) [^<>]*?style=[^<>]*?FONT-SIZE: [0-9.]+[^\<>]*?>/img,
function (){notv="|select|input|option|object|"
if(notv.indexOf("|"+arguments[1].toLowerCase()+"|")==-1){
op=mio.length
mio[op]=arguments[0]
return "<opis"+op+">"}else{return arguments[0]}})
kba=["h1","h2","h3","h4","h5","h6","pre","button","listing","big","small","tt","
code","kbd","samp","td","\\td","caption","\\caption","th","\\th","tr","\\tr","table","\\t
able","thead","\\thead","tbody","\\tbody","tfoot","\\tfoot"]
for(b in kba){
fd="<("+kba[b]+") [^\<>]*?>"
jk=new RegExp(fd,["img"])
partB=partB.replace(jk,
function (){op=mio.length
mio[op]=arguments[0]
return "<opis"+op+">"}
)
}
//收尾工作
liming=[]
partB=partB.replace(/<([opis|site|lonf])([0-9]*)>/g,

```

```

function(){var op=liming.length
if(arguments[1]=="opis"){liming[op]=mio[parseInt(arguments[2])]}
if(arguments[1]=="site"){liming[op]="<font"+dol[parseInt(arguments[2])]+>" }
if(arguments[1]=="lonf"){liming[op]="</font>" }
return "<duan"+op+">"
})
if(liming.length>0){
partB=partB.replace(/^(.+?)(<duan0>)/m,function(){if("!="arguments[1]){return
addpre+arguments[1]+"</font>" +arguments[2]}}
else{return arguments[0]}})
var op=liming.length-1
for(var kc=0;kc<op;kc++){
fd="(<duan"+kc+">)(.+?)(<duan"+(kc+1)+">)"
jk=new RegExp(fd,["m"])
partB=partB.replace(jk,
function(){if("!="arguments[2]){return arguments[1]+addpre+arguments[2]+ "</font>" +
arguments[3]}
else{return arguments[0]}})})
fd="(<duan"+op+">)(.+?)$"
jk=new RegExp(fd,["m"])
partB=partB.replace(jk,function(){if("!="arguments[2]){return arguments[1] +addpre+
arguments[2]+"</font>" }
else{return arguments[0]}})})
}
else{partB=addpre+partB+"</font>"
}
partB=partB.replace(/<duan([0-9]+)>/g,function(){return liming[parseInt (arguments [1])])})
var endtemp=partA+partB+partC
for(vof in cook)endtemp=endtemp.replace("<cook"+vof+">","cook["+vof])
Editor.document.body.innerHTML=endtemp
var vrvd=Editor.document.selection.createRange()
vrvd.moveToBookmark(sBookmark)
vrvd.select()
}
else{
Editor.document.selection.createRange().pasteHTML(addpre+"</font>")
}Editor.focus()
el.blur()
el.selectedIndex=0}
</script>

```

需要在 body 中添加多选框，用来选择字体、字号和样式等，具体内容可参考随书光盘。

【运行效果】

文本编辑器的运行效果如图 3-37 所示。

【难点剖析】

本例的重点是字体的获取，以及执行命令“execCommand”。字体获取使用的是方法

“getSystemFonts”，方法中的“dlgHelper”对象必须在页面中注册，本例中注册的代码是“<OBJECT id=dlgHelper CLASSID="clsid:3050f819-98b5-11cf-bb82-00aa00bdce0b" width="0px" height="0px"></OBJECT>”。ExecCommand 是文本编辑器中的一个重要方法，主要用于样式的变更。

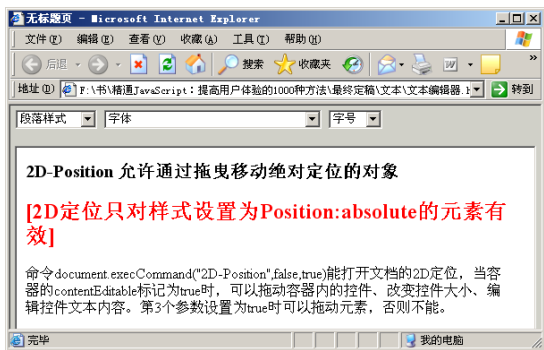


图 3-37 文本编辑器的运行效果

第 4 章 鼠标特效

本章导读

鼠标是用户操作页面的主要工具，用户可以使用鼠标滚动页面、选择控件，还可以使用鼠标在页面中画图。本章介绍如何捕获页面中的鼠标事件，以及常见的一些鼠标操作技巧。

4.1 禁用鼠标右键

【实例描述】

为了保证网站的内容不被非法拷贝，可以通过判断鼠标按键来禁止用户操作。本例禁止用户使用鼠标右键。

【实现代码】

```
<script language="javascript">
function click()
{
    if (event.button==2)                //单击的鼠标键为右键
    {
        alert('本网站禁用右键');
    }
}
document.onmousedown=click;           //绑定事件
</script>
```

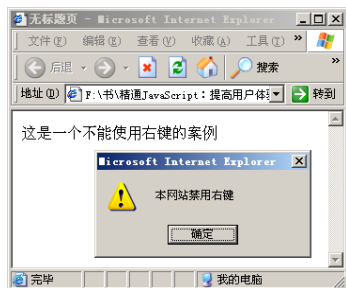


图 4-1 禁用鼠标右键的提示效果

【运行效果】

禁用鼠标右键的提示效果如图 4-1 所示。

【难点剖析】

本例有两个重点：捕获窗体的鼠标事件和判断用户单击的按键。“onmousedown”事件用来捕获窗体的鼠标事件；“event.button”判断用户单击的按键，“1”表示左键，“2”表示右键。

4.2 使鼠标滚轮失效

【实例描述】

当页面内容太多时，可以通过鼠标滚轮实现翻页效果。但有些读书网站提供页面自动滚屏的功能，此时可使鼠标的滚轮失效，让用户充分体验网站提供的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function document.onmousewheel()      //滚轮事件重新定义
{
    return false;                      //返回 false 表示什么都不操作
}
```

```

</script>
</head>
<body>
  <table style="width: 300px">
    <tr>
      <td>
        阿</td>
      <td>
        不</td>
      <td>
        才</td>
    </tr>
    <tr>
      <td>
        的</td>
      <td>
        恶</td>
      <td>
        发</td>
    </tr>
    <tr>
      <td>
        的</td>
      <td>
        发</td>
      <td>
        的</td>
    </tr>
  </table>
</body>
</html>

```

【难点剖析】

本例在页面中添加了一个表格，可将窗口缩小，以出现滚动条效果。当出现滚动条时，滑动鼠标的滚轮，测试此操作是否能够成功。代码重写了“document.onmousewheel”方法，若返回 false 值则表示不执行任何鼠标滚动的操作。

4.3 状态栏显示鼠标位置

【实例描述】

为了让 DIV 层可以跟随鼠标，层的控制通常需要获取鼠标的位置。本例学习如何获取鼠标的 X 和 Y 坐标。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head>
<title>标题页</title>
<script language=javascript>
//移动鼠标的方法
function move(e){
if (document.layers) { var x=e.pageX; var y=e.pageY;}
if (document.all) { var x=event.x;
var y=event.y+document.body.scrollTop;}
status="x:"+x+" y:"+y;           //组合鼠标的(x,y)坐标
}
document.onmousemove = move;      //鼠标移动时绑定 move 方法
if (document.layers) document.captureEvents(Event.MOUSEMOVE);
</script>
</head>
<body>
</body>
</html>
```

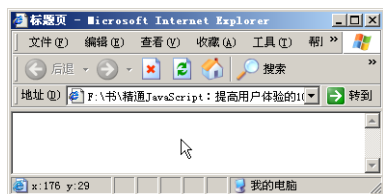


图 4-2 状态栏的显示效果

条。Event 对象的“y”属性表示 y 坐标。

【运行效果】

状态栏的显示效果如图 4-2 所示。

【难点剖析】

本例的重点 Event 对象，其用来描述 JavaScript 的事件。Event 对象的“x”属性用来获取鼠标指针位置相对于窗口客户区域的 x 坐标，其中客户区域不包括窗口自身的控件和滚动

4.4 单击鼠标右键到指定页

【实例描述】

在网页上单击右键，会弹出 IE 自带的一些菜单命令，如复制、查看源代码等。为了屏蔽这些功能，可以在用户单击右键时，实现页面的导航。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=JavaScript>
//判断浏览器的类型
if (navigator.appName.indexOf("Internet Explorer") != -1)
    document.onmousedown = newPage;
function newPage()
{
    if (event.button == 2 )
    {
```



```

        alert("即将导航到搜索页!");
        location.replace("http://google.com");
    }
}
</script>
</head>
<body>
</body>
</html>

```

【难点剖析】

本例的难点是浏览器类型的判断和鼠标右键的判断。通过检查浏览器名称中是否包含“Internet Explorer”字符来判断浏览器的类型。通过“event.button”判断用户单击了哪个键，“1”表示左键，“2”表示右键。

4.5 鼠标放到图片上会显示另外一张图片

【实例描述】

在大多数网页中，如果鼠标移动到图片上，通常会给出文本提示。本例将学习如何不显示文本，而显示指定的图片。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function changeSrc(obj)
{
    obj.src="LOGO2.gif";           //改变图像地址
}
</script>
</head>
<body>

</body>
</html>

```

【运行效果】

页面初始的效果如图 4-3 所示。鼠标移到图片上的效果如图 4-4 所示。

【难点剖析】

本例的重点是如何判断当前图片，并更改图片的地址。在控件中，“src”属性表示要显示的图片的地址，如果是网络地址，一定要指明详细的 URL 路径。如果判断的是当前图片，在参数传递的时候，使用“this”。



图 4-3 页面初始的效果

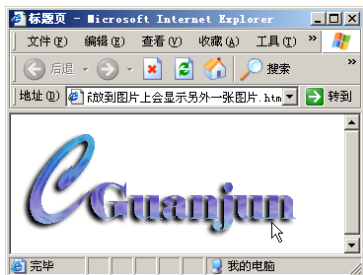


图 4-4 鼠标移到图片上的效果

4.6 鼠标形状定义大全

【实例描述】

在网页的不同状态下，鼠标可以展示为不同的样式，本例罗列了常用的鼠标样式。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<table>
<tr><td><a href="#" style="cursor:auto">随机自动箭头</a></td><td><a href="#" style=
"cursor:default">标准箭头</a></td></tr>
<tr><td><a href="#" style="cursor:hand">手形光标</a></td><td><a href="#" style=
"cursor:wait">等待光标</a></td></tr>
<tr><td><a href="#" style="cursor:text">I 形光标</a></td><td><a href="#" style="cursor:
vertical-text">水平 I 形光标</a></td></tr>
<tr><td><a href="#" style="cursor:no-drop">不可拖动光标</a></td><td><a href="#" style=
"cursor:not-allowed">无效光标</a></td></tr>
<tr><td><a href="#" style="cursor:help">?帮助光标</a></td><td><a href="#" style=
"cursor:all-scroll">三角方向标</a></td></tr>
<tr><td><a href="#" style="cursor:move">移动标</a></td><td><a href="#" style="cursor:
crosshair">十字标</a></td></tr>
</table>
</body>
</html>
```

【运行效果】

鼠标样式的展示效果如图 4-5 所示。

【难点剖析】

本例中的重点是 CSS 中鼠标样式的设置。如果要设置鼠标停留在某控件上的样式，则在此控件的 style 上，设置其“cursor”属性。

4.7 鼠标移入移出时颜色变化

【实例描述】

当用户将鼠标移动到按钮或链接上时，为了突出显示用户的选择，会改变链接和按钮的颜色。默认链接的颜色是自动变化的，本例学习如何手动为按钮设置这种效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type="submit" value="把鼠标移动到这里" name="btn1" onMouseOut=this.style.color=
"blue"
onMouseOver=this.style.color="red" >
</body>
</html>
```

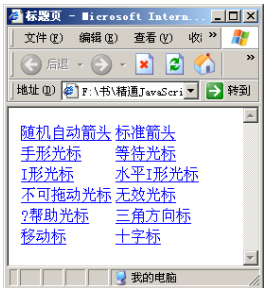


图 4-5 鼠标样式的展示效果

【难点剖析】

本例的重点是鼠标事件的应用。当鼠标移动到控件上时，触发了“onMouseOver”事件，当鼠标移走时，触发“onMouseOut”事件。两个事件的响应都是使用“style.color”，改变控件的颜色。

4.8 跟随鼠标的文字

【实例描述】

这是最常见的文本特效，不管鼠标移动到什么地方，都有一段文字跟随其后。本例将学习此特效的制作。

【实现代码】

```
<style type="text/css">
.spanstyle {
position:absolute;
visibility:visible;
top:-50px;
font-size:9pt;
color: #000000;
font-weight:bold;}
</style>
<script language="javascript">
var x,y;
var step=20;
```



```

var flag=0;
var message="我就追随鼠标。";
message=message.split(""); //将文本切割成数组
var xpos=new Array();
for (i=0;i<=message.length-1;i++) {xpos[i]=-50;}
var ypos=new Array();
for (i=0;i<=message.length-1;i++) {ypos[i]=-50;}
function handlerMM(e)
{
//判断浏览器，同时获取鼠标的坐标
x = (document.layers) ? e.pageX : document.body.scrollLeft+event.clientX;
y = (document.layers) ? e.pageY : document.body.scrollTop+event.clientY;
flag=1;
}
function makesnake()
{
if (flag==1 && document.all)
{
for (i=message.length-1; i>=1; i--) {
xpos[i]=xpos[i-1]+step;
ypos[i]=ypos[i-1];      }
xpos[0]=x+step;          //文本的 X 坐标距离鼠标 X 坐标的距离
ypos[0]=y;               //文本和鼠标的 Y 坐标相同
//设置包装文本的 span 控件的位置
for (i=0; i<message.length-1; i++) {
var thisspan = eval("span"+(i)+"."+style");
thisspan.posLeft=xpos[i];
thisspan.posTop=ypos[i];    } }
else if (flag==1 && document.layers) {
for (i=message.length-1; i>=1; i--) {
xpos[i]=xpos[i-1]+step;
ypos[i]=ypos[i-1];      }
xpos[0]=x+step;
ypos[0]=y;
for (i=0; i<message.length-1; i++) {
var thisspan = eval("document.span"+i);
thisspan.left=xpos[i];
thisspan.top=ypos[i];    }
var timer=setTimeout("makesnake()",30);}
//显示文本的重点
for (i=0;i<=message.length-1;i++)
{
document.write("<span id='span"+i+"'class='spanstyle'>");
document.write(message[i]);
document.write("</span>");
}
//针对 navigater 浏览器时的情况
if (document.layers){

```

```
document.captureEvents(Event.MOUSEMOVE);}
document.onmousemove = handlerMM;           //将方法绑定到鼠标的移动事件
</script>
```

需要在 body 的加载事件中，调用上面的方法，代码如下所示：

```
<body onLoad="makesnake()">
```

【运行效果】

跟随鼠标的文字效果如图 4-6 所示。

【难点剖析】

本例的难点是如何获取鼠标的(x, y)坐标，如何设置文字的坐标跟随鼠标的坐标。移动时文字的延迟通过“setTimeout”定时器解决。

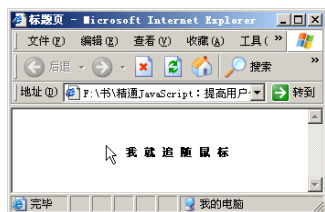


图 4-6 跟随鼠标的文字效果

4.9 跟随鼠标的彩色文字

【实例描述】

跟随鼠标的文本可以很轻易地实现，本例要实现的是文字不仅要跟随鼠标，而且颜色会不断变化。

【实现代码】

```
<SCRIPT language=javascript>
    var message="欢迎参加北京奥运会!";           //要跟随鼠标的文本
    var x,y;
    var step=15;                                     //文本之间的距离
    var flag=0;

    message=message.split("");                       //将文本切换成数组
    var xpos=new Array();
    for (i=0;i<=message.length-1;i++) {             //设置所有文本的初始 x 坐标
        xpos[i]=-50;
    }
    var ypos=new Array();
    for (i=0;i<=message.length-1;i++) {             //设置所有文本的初始 y 坐标
        ypos[i]=-50;
    }
    function handlerXY(e) {
        //根据浏览器的不同，获取鼠标的(x,y)坐标
        x = (document.layers) ? e.pageX : document.body.scrollLeft+event.clientX+10;
        y = (document.layers) ? e.pageY : document.body.scrollTop+event.clientY;
        flag=1;
    }
    function makesnake() {
        if (flag==1 && document.all) {
```



```

        for (i=message.length-1; i>=1; i--) {
            xpos[i]=xpos[i-1]+step;           //设置文本之间的显示距离
            ypos[i]=ypos[i-1];                 //设置文本的 y 坐标
        }

        xpos[0]=x+step;
        ypos[0]=y;
        for (i=0; i<=message.length-1; i++) {
            var thisspan = eval("span"+(i)+"style");
            thisspan.posLeft=xpos[i];
            thisspan.posTop=ypos[i];
            thisspan.color=Math.random() * 255 * 255 * 255 + Math.random() * 255
* 255 + Math.random() * 255;
        }
    }
    else if (flag==1 && document.layers) {
        for (i=message.length-1; i>=1; i--) {
            xpos[i]=xpos[i-1]+step;
            ypos[i]=ypos[i-1];
        }
        xpos[0]=x+step;
        ypos[0]=y;

        for (i=0; i<message.length-1; i++)           //根据字符的个数创建多个 span 元素, //
        用来显示字符
        {
            var thisspan = eval("document.span"+i);
            thisspan.left=xpos[i];                     //指定 span 的 x 坐标
            thisspan.top=ypos[i];                       //指定 span 的 y 坐标
                                                    //创建随机的颜色值
            thisspan.color=Math.random() * 255 * 255 * 255 + Math.random() * 255
* 255 + Math.random() * 255;
        }
    }
}
for (i=0;i<=message.length-1;i++) {                //循环输出文本信息
    document.write("<span id='span"+i+"' class='spanstyle'>");
    document.write(message[i]);
    document.write("</span>");
}

if (document.layers) {
    document.captureEvents(Event.MOUSEMOVE);
}
document.onmousemove = handlerXY;
function firstLoad() {                             //窗体一加载便触发的事件
    makesnake();                                     //实现跟随鼠标文本的方法
    window.setTimeout("firstLoad();", 5);           //定时执行此方法
}

```

</SCRIPT>

【运行效果】

跟随鼠标的彩色文本效果如图 4-7 所示。

【难点剖析】

本例的重点如下：

(1) 动态输出 span 元素，用来显示文本内容。动态输出 span 元素使用的是“document.write”方法，此处要注意每个“span”元素 ID 的创建方式。

(2) 获取鼠标的坐标，根据坐标位置显示文本信息。获取鼠标坐标使用的是“event.clientX”和“event.clientY”属性。

(3) 动态实现颜色的变化。本例使用了“Math.random() * 255 * 255 * 255”创建一个随机颜色值，然后使用 span 元素的“color”属性设置文本最后显示的颜色。



图 4-7 跟随鼠标的彩色文本效果

4.10 跟随鼠标的魔法文字

【实例描述】

鼠标周围可以跟随图片、文字和星星等。本例将提供一些类似对联的文字，一直跟随鼠标移动。

【实现代码】

```
<SCRIPT language=JavaScript1.2>
var msg='看的见我的漂移吗?';
var msgColor='000000'
var dismissafter=0
var amount=5,ypos=0,xpos=0,Ay=0,Ax=0,By=0,Bx=0,Cy=0,Cx=0,Dy=0,Dx=0,Ey=0,Ex=0;
if (document.layers){ //Netscape 下
    for (i = 0; i < amount; i++)
        {document.write('<layer name=ns'+i+' top=0 left=0><font face="宋体" size=2 color=
'+msgColor+'>'+msg+'</font></layer>')}
    window.captureEvents(Event.MOUSEMOVE);
    function nsmouse(evt) //通过鼠标事件，获取鼠标的坐标
        {xpos = evt.pageX;ypos = evt.pageY;mouseFollow()}
}
else if (document.all){ //IE 下——输出层
    document.write("<div id='outer' style='position:absolute;top:0px;left: 0px'>");
    document.write("<div id='inner' style='position:relative'>");
    for (i = 0; i < amount; i++)
        {document.write('<div id="text"'+i+' style="position:absolute;top:0px;left:
0px;font-family:Courier New;font-size: 16px;color:'+msgColor+'">'+msg+'</div>')}
    document.write("</div>");
    document.write("</div>");
    function iemouse()
```

//通过鼠标事件，获取鼠标的坐标



```

        {ypos = document.body.scrollTop + event.y;xpos = document.body.scrollLeft + event.
x;mouseFollow()}}
    }
    function mouseFollow(){
        if (document.layers){//Netscape 下——设置层的位置
            document.layers["ns0"].top=ay;document.layers["ns0"].left=ax;
            document.layers["ns1"].top=by;document.layers["ns1"].left=bx;
            document.layers["ns2"].top=cy;document.layers["ns2"].left=cx;
            document.layers["ns3"].top=Dy;document.layers["ns3"].left=Dx;
            document.layers["ns4"].top=Ey;document.layers["ns4"].left=Ex;
        }
        else if (document.all){//IE 下，设置层的位置
            outer.all.inner.all[0].style.pixelTop=ay;outer.all.inner.all[0].style. pixelLeft =ax;
            outer.all.inner.all[1].style.pixelTop=by;outer.all.inner.all[1].style. pixelLeft =bx;
            outer.all.inner.all[2].style.pixelTop=cy;outer.all.inner.all[2].style. pixelLeft =cx;
            outer.all.inner.all[3].style.pixelTop=Dy;outer.all.inner.all[3].style. pixelLeft =Dx;
            outer.all.inner.all[4].style.pixelTop=Ey;outer.all.inner.all[4].style. pixelLeft =Ex;
        }
    }
    function move(){
        if (dismissafter!=0)
            setTimeout("hideMove()",dismissafter*1000)
        if (document.layers){window.onMouseMove = nsmouse}
        else if (document.all){window.document.onmousemove = iemouse} //绑定鼠标事件
        ey = Math.round(Ey+=((ypos+20)-Ey)*2/2);ex = Math.round(Ex+=((xpos+20) -Ex)*2/2);
        dy = Math.round(Dy+=(ey - Dy)*2/4);dx = Math.round(Dx+=(ex - Dx)*2/4);
        cy = Math.round(Cy+=(dy - Cy)*2/6);cx = Math.round(Cx+=(dx - Cx)*2/6);
        by = Math.round(By+=(cy - By)*2/8);bx = Math.round(Bx+=(cx - Bx)*2/8);
        ay = Math.round(Ay+=(by - Ay)*2/10);ax = Math.round(Ax+= (bx - Ax)*2/10);
        mouseFollow();
        jumpstart=setTimeout('move()',10); //定时执行捕获操作
    }
    function hideMove(){
        if (document.all){ //IE 浏览器的情况下
            for (i2=0;i2<amount;i2++){
                outer.all.inner.all[i2].style.visibility="hidden" //设置为隐藏
                clearTimeout(jumpstart) //清除定时器
            }
        }
        else if (document.layers){ //Netscape 浏览器的情况下
            for (i2=0;i2<amount;i2++){
                temp="ns"+i2
                document.layers[temp].visibility="hide"
                clearTimeout(jumpstart)
            }
        }
    }
    window.onload=move; //窗体一加载便触发飘移事件

```


</SCRIPT>

【运行效果】

魔法文字跟随鼠标的效果如图 4-8 所示。

【难点剖析】

本例的重点是获取鼠标的活动坐标，根据坐标动态设置多个层，多个层的交替形成魔法效果。“event.x”和“event.y”分别获取鼠标的 x 和 y 坐标，然后使用“document.onmousemove= iemouse”绑定鼠标的移动事件。

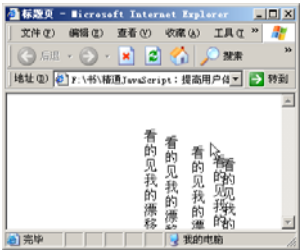


图 4-8 魔法文字跟随鼠标的效果

4.11 跟随鼠标的星星

【实例描述】

为了增加鼠标的显示效果，本例在鼠标周围布置一些彩色的星星，这些星星会跟随鼠标移动，而且实现闪烁效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
看看鼠标周围的效果，是不是五颜六色的星星。
<script language="javascript">
function Div_Layer(divleft,divtop,divfnx,divfny,mydiv,divbilder,divloop, divto,
divcnt,divstep)
{
if ((document.layers)||(document.all)){
with (Math) {yynextx= eval(divfnx)} //设置为数值型数据
with (Math) {yynexty= eval(divfny)}
divcnt=(divloop && divcnt>=divstep*divbilder)?0:divcnt+divstep;
if (document.layers){
eval(mydiv+".top="+ (yynexty+divtop)) //Netscape 下设置横坐标和纵坐标
eval(mydiv+".left="+ (yynextx+divleft))
}
if (document.all){
eval("mydiv=mydiv.replace(/.layers/gi, '.all')");
eval(mydiv+".style.pixelTop="+ (yynexty+divtop)); //设置div 的横坐标
eval(mydiv+".style.pixelLeft="+ (yynextx+divleft)); //设置div 的纵坐标
}
argStr='Div_Layer('+divleft+', '+divtop+', '+divfnx+', '+divfny+', '+mydiv+',
'+divbilder+', '+divloop+', '+divto+', '+divcnt+', '+divstep+')';
if (divcnt<=divstep*divbilder){
eval(mydiv+".divto=setTimeout(argStr,divto)"); //设置定时器——实现星星的闪烁效果
```



```

    }
  }
}
function YY_Mousetrace(evnt) //设置鼠标移动的事件
{
  if (yyins4) //Netscape 浏览器的情况
  {if (evnt.pageX) {yy_ml=evnt.pageX; yy_mt=evnt.pageY;} }
  else{
    yy_ml=(event.clientX + document.body.scrollLeft); //横坐标位置
    yy_mt=(event.clientY + document.body.scrollTop); //纵坐标位置
  }
  if (yy_tracescript)eval(yy_tracescript) //转换为数值型变量
}
</script>
<div id="tooltip2"
  style="position:absolute;visibility:hidden;clip:rect(0 150 50 0);width:150px;
background-color:lightyellow">
  <layer name="nstip" width="1000px" bgcolor="lightyellow"> </layer>
</div>
  <div id="yyd0"
    style="position:absolute; left:10px; top:50px; width:3px; height:3px; z-index:1;
background-color: #19636c; layer-background-color: #19636c; border: 1px none #000000; clip:
rect(0 3 3 0)"></div>
    <div
      id="yyd1"
      style="position:absolute; left:20px; top:50px; width:3px; height:3px; z-index:1;
background-color: #708574; layer-background-color: #708574; border: 1px none #000000; clip:
rect(0 3 3 0)"></div>
      <div
        id="yyd2"
        style="position:absolute; left:30px; top:50px; width:3px; height:3px; z-index:1;
background-color: #379bbf; layer-background-color: #379bbf; border: 1px none #000000; clip:
rect(0 3 3 0)"></div>
        <div
          id="yyd3"
          style="position:absolute; left:40px; top:50px; width:3px; height:3px; z-index:1;
background-color: #25184c; layer-background-color: #25184c; border: 1px none #000000; clip:
rect(0 3 3 0)"></div>
          <div
            id="yyd4"
            style="position:absolute; left:50px; top:50px; width:3px; height:3px; z-index:1;
background-color: #31bd3c; layer-background-color: #31bd3c; border: 1px none #000000; clip:
rect(0 3 3 0)"></div>
            <div
              id="yyd5"
              style="position:absolute; left:60px; top:50px; width:3px; height:3px; z-index:1;
background-color: #c11efd; layer-background-color: #c11efd; border: 1px none #000000; clip:
rect(0 3 3 0)"></div>

```

```

<script language="javascript">
    var yyns4=window.Event?true:false; var yy_mt = 0; var yy_ml = 0;
                                                //点的坐标
    document.onmousemove = YY_Mousetrace;      //绑定鼠标移动的事件
    yy_tracescript = '';
    Div_Layer(0,0,'yy_ml+cos((15*sin(divcnt/39.83007847812662))+0)*150* (sin(10+
divcnt/20)+0.2)*cos(divcnt/20)','yy_mt+sin((15*sin(divcnt/34.224861639800686))+0)*150*(si
n(10+divcnt/20)+0.2)*cos(divcnt/20)','document.layers[\`'yyd0\`]',2000,true,80,0,1);
    Div_Layer(0,0,'yy_ml+cos((15*sin(divcnt/27.66510707209673))+30)*150*
(sin(10+divcnt/20)+0.2)*cos(divcnt/20)','yy_mt+sin((15*sin(divcnt/9.240632767417667))+30)*
150*(sin(10+divcnt/20)+0.2)*cos(divcnt/20)','document.layers[\`'yyd1\`]',2000,true,80,0,1);
    Div_Layer(0,0,'yy_ml+cos((15*sin(divcnt/16.45318944579641))+60)*150* (sin(10+
divcnt/20)+0.2)*cos(divcnt/20)','yy_mt+sin((15*sin(divcnt/16.0564452288292))+60)*150*( si
n(10+divcnt/20)+0.2)*cos(divcnt/20)','document.layers[\`'yyd2\`]',2000,true,80,0,1);
    Div_Layer(0,0,'yy_ml+cos((15*sin(divcnt/6.95348954836835))+90)*150*
(sin(10+divcnt/20)+0.2)*cos(divcnt/20)','yy_mt+sin((15*sin(divcnt/44.13697049887155))+90)*
150*(sin(10+divcnt/20)+0.2)*cos(divcnt/20)','document.layers[\`'yyd3\`]',2000,true,80,0,1);
    Div_Layer(0,0,'yy_ml+cos((15*sin(divcnt/33.90077294583733))+120)*150* (sin(10+
divcnt/20)+0.2)*cos(divcnt/20)','yy_mt+sin((15*sin(divcnt/2.2378828869411587))+120)*150*(
sin(10+divcnt/20)+0.2)*cos(divcnt/20)','document.layers[\`'yyd4\`]',2000,true,80,0,1);
    Div_Layer(0,0,'yy_ml+cos((15*sin(divcnt/37.858312521039835))+150)*150*(sin(10+divcnt/20)+
0.2)*cos(divcnt/20)','yy_mt+sin((15*sin(divcnt/18.083839795990098))+150)*150*(sin(10+divc
nt/20)+0.2)*cos(divcnt/20)','document.layers[\`'yyd5\`]',2000,true,80,0,1);

</script>
</body>
</html>

```

【运行效果】

跟随鼠标的星星如图 4-9 所示。

【难点剖析】

本例的重点是这些星星的布局、颜色和跳动。星星通过 div 实现，颜色通过设置 div 的样式“background-color”实现，有多少颗星星，就有多少个 div。为了实现星星的五颜六色效果，每个 div 的颜色都不相同。最后通过一个“setTimeOut”定时器，实现星星的跳动效果。

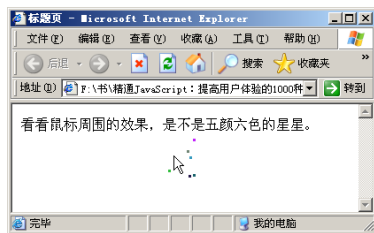


图 4-9 跟随鼠标的星星

4.12 跟随鼠标的旋转背景

【实例描述】

鼠标在页面中移动的时候，会一直有一个旋转的椭圆点跟随它。本例学习如何实现这种效果。

【实现代码】

```

<SCRIPT LANGUAGE="JavaScript">
    var Clrs = new Array(6);                                //随机的背景颜色，保存在一个数组中

```



```

    Clrs[0] = 'ff0000';
    Clrs[1] = '00ff00';
    Clrs[2] = '000aff';
    Clrs[3] = 'ff00ff';
    Clrs[4] = 'fff000';
    Clrs[5] = 'fffff0';
    var yBase = 200;
    var xBase = 200;
    var step;
    var currStep = 0;
    var Xpos = 1; //横坐标变量
    var Ypos = 1; //纵坐标变量
    if (document.all) {
        function MoveHandler()
        {
            Xpos = document.body.scrollLeft+event.x; //设置横坐标
            Ypos = document.body.scrollTop+event.y; //设置纵坐标
        }
        document.onmousemove = MoveHandler; //绑定鼠标的移动事件
    }
    function Comet()
    {
        if (document.all) {
            yBase = window.document.body.offsetHeight / 4;
            xBase = window.document.body.offsetWidth / 4;
        }
        if (document.all) {
            for ( i = 0 ; i < starsDiv.all.length ; i++ ) { //循环设置每个 div 的显示位置
                step = 3;
                starsDiv.all[i].style.top = Ypos + yBase*Math.cos((currStep + i*4)/12)*Math.
cos(0.7+currStep/200);
                starsDiv.all[i].style.left = Xpos + xBase*Math.sin((currStep + i*3)/10)*Math.
sin(8.2+currStep/400);
                for (ai = 0; ai < Clrs.length; ai++) {
                    var c=Math.round(Math.random()*[ai]); //获取随机的一个颜色值
                }
                starsDiv.all[i].style.background = Clrs[c]; //动态设置 div 的背景色
            }
        }
        currStep += step;
        setTimeout("Comet()", 5); //设置定时器——实现旋转效果
    }
    Comet();
</script>

```

【运行效果】

跟随鼠标的旋转背景如图 4-10 所示。

【难点剖析】

本例的重点是旋转背景所在的层。跟随鼠标的旋转背景由多个不同颜色的点组成，其中每个点是一个层（div），每个层的颜色不同。这些层的位置通过鼠标的坐标获取，此处要注意层的位置使用“position:relative”，表示相对坐标，其坐标的原始点是这些层的父级节点“starsDiv”。

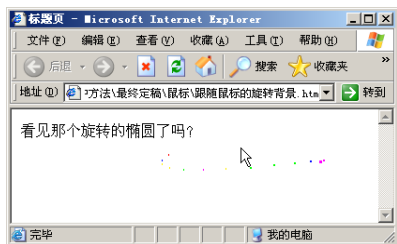


图 4-10 跟随鼠标的旋转背景

4.13 图片跟随鼠标

【实例描述】

要在网页中捕获鼠标的位置，需要借助于鼠标的移动事件。本例将通过绑定鼠标的移动事件，为鼠标提供一个特效，让图片一直跟随。

【实现代码】

```
<SCRIPT LANGUAGE="JavaScript">
    var newtop=0;
    var newleft=0;
    divStyleRef="layer.style.";
    layerRef="document.all";
    mystyle=".style";
    function doMouseMove()
    {
        layerName = 'myimg'; //获取包装图片的div
        eval('var curElement='+layerRef+'['+layerName+']');
        eval(layerRef+'['+layerName+']'+mystyle+'.visibility="hidden"');
        eval('curElement'+mystyle+'.visibility="visible"');
        //设置div的显示位置——坐标和高度、宽度
        eval('newleft=document.body.clientWidth-curElement'+mystyle+'.pixelWidth');
        eval('newtop=document.body.clientHeight-curElement'+mystyle+'.pixelHeight');
        eval('height=curElement'+mystyle+'.height');
        eval('width=curElement'+mystyle+'.width');
        width=parseInt(width);
        height=parseInt(height);
        //鼠标移出边界时的div位置的判断
        if (event.clientX > (document.body.clientWidth - 5 - width))
        {
            newleft=document.body.clientWidth + document.body.scrollLeft - 5 - width;
        }
        else
        {
            newleft=document.body.scrollLeft + event.clientX;
        }
        eval('curElement'+mystyle+'.pixelLeft=newleft');
        if (event.clientY > (document.body.clientHeight - 5 - height))
```



```
{
    newtop=document.body.clientHeight + document.body.scrollTop - 5 - height;
}
else
{
    newtop=document.body.scrollTop + event.clientY;
}
eval('curElement'+mystyle+'.pixelTop=newtop');
}
document.onmousemove = doMouseMove;           //绑定鼠标移动事件
                                              //动态输出 div 和图像——默认情况下图像隐藏

document.write('<div ID=mydiv>');
document.write('')
document.write('</div>')
</SCRIPT>
```

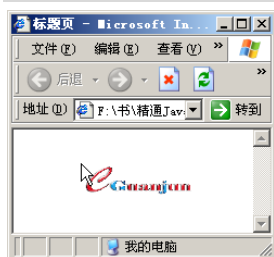


图 4-11 图片跟随鼠标的效果

【运行效果】

图片跟随鼠标的效果如图 4-11 所示。

【难点剖析】

本例的难点在于如何捕获鼠标的位置。本例通过 `event.clientX` 和 `event.clientY` 获取鼠标的 X 和 Y 坐标，然后设置图像的隐藏和显示，以达到图片跟随鼠标的效果。

4.14 围绕鼠标的文本

【实例描述】

文本可以跟随鼠标，也可以围绕着鼠标闪动。本例将演示如何让指定的文本，一直围绕着鼠标闪动。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT language=javascript>
var cx=0;
var cy=0;
var val=0;
function location()
{
    cx=window.event.x;           //获取鼠标的 x 坐标
    cy=window.event.y           //获取鼠标的 y 坐标
}
document.onmousemove=location;  //绑定鼠标的移动事件
function follow(i)
```

```

{
    var x;
    if(i<4)x=cx-50+i*10;                //设置要显示的字符的(x,y)坐标
    else x=cx-25+i*10;
    var y=cy-20+Math.floor(Math.random()*40);    //实现随机值的获取
    w=eval("word"+i);                    //获取要显示的字符
    with(w.style)
    {
        left=x.toString()+"px";        //在指定的位置显示字符
        top=y.toString()+"px";
    }
}
function show(i)
{
    var w=eval("word"+i);                //获取要显示的字符
    with(w.style)
    {
        visibility="visible";
        s=parseInt(fontSize);            //设置字符的字体——从大到小,从小到大
        if(s>=200)s-=100;
        else if(s>90&&s<=100)
        {
            s-=85;
            clearInterval(val);
            if(i<5)val=setInterval("show('"+(i+1)+"')",20); //循环地显示下一个字符
        }
        fontSize=s;
    }
}
function start()
{
    for(i=1;i<=5;i++)
    {
        val=setInterval("show(1)",20);    //循环执行
        setInterval("follow('"+i+"')",100); //循环执行
    }
}
</SCRIPT>
<SCRIPT language=javascript>
    var word=new Array(5);
    word[1]="欢";word[2]="迎";word[3]="您";word[4]="光";word[5]="临"; //设置要围绕的字符数组
    for(i=1;i<=5;i++)
        document.write("<div id='word"+i+"' style='width:20px;height:20px;position:
absolute;font-size:1000;visibility:hidden'><font face='Forte'
color='#cc9966'>"+word[i]+"</font></div>");
    start();                                //开始执行围绕操作
</SCRIPT>
</head>
<body>
</body>

```

</html>

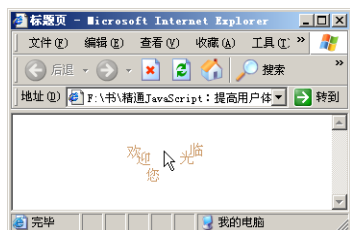


图 4-12 文字围绕鼠标的效果

【运行效果】

文字围绕鼠标的效果如图 4-12 所示。

【难点剖析】

本例中围绕鼠标的文字颜色是固定的，位置和字体是不断变化的。通过数组“word”保存要显示的文字，然后用“show”方法逐个显示数组中的文本，“follow”方法用来设置这些文字的随机位置。

4.15 鼠标旁边的提示信息

【实例描述】

在大型门户网站中，为了让用户可以全面地了解网站的内容，通常需要简化很多按钮，但为了让用户可以了解按钮的用途，当鼠标指向按钮时，会出现提示信息详细介绍按钮的功能。本例介绍如何实现鼠标旁边的提示信息。

【实现代码】

```
<script Language="javascript">
//内部变量定义
tPopWait=50;
tPopShow=4000;
showPopStep=15;
popOpacity=80;
sPop=null;
curShow=null;
tFadeOut=null;
tFadeIn=null;
tFadeWaiting=null;
//动态显示提示信息，注意此处定义了样式和层
document.write("<style type='text/css' id='defaultPopStyle'>");
document.write(".cPopText { background-color: #F8F8F5;color:#000000; border: 1px
#000000 solid;font-color: font-size: 12px;padding-right: 4px;padding-left: 4px;height: 20px;
padding-top: 2px;padding-bottom: 2px; filter: Alpha(Opacity=0)}");
document.write("</style>");
document.write("<div id='dypopLayer' style='position:absolute;z-index: 1000;'
class='cPopText'></div>");

//显示提示信息的方法
function showPopupText()
{
    var o=event.srcElement;                //获取鼠标指向的链接或按钮
    MouseX=event.x;
    MouseY=event.y;
```



```

if(o.alt!=null && o.alt!="")
{o.dypop=o.alt;o.alt=""};
if(o.title!=null && o.title!="")
{o.dypop=o.title;o.title=""};
if(o.dypop!=sPop)
{
    sPop=o.dypop;                                //设置提示信息的内容——从控件的 title 属性获得
    clearTimeout(curShow);
    clearTimeout(tFadeOut);
    clearTimeout(tFadeIn);
    clearTimeout(tFadeWaiting);
    if(sPop==null || sPop=="")
    {
        dypopLayer.innerHTML="";
        dypopLayer.style.filter="Alpha()";
        dypopLayer.filters.Alpha.opacity=0;
    }
    else
    {
        if(o.dyclass!=null)
        popStyle=o.dyclass
        else popStyle="cPopText";
        curShow=setTimeout("showIt()",tPopWait);
    }
}
}
//定义显示的位置
function showIt()
{
    dypopLayer.className=popStyle;
    dypopLayer.innerHTML=sPop;
    popWidth=dypopLayer.clientWidth;
    popHeight=dypopLayer.clientHeight;
    if(MouseX+12+popWidth>document.body.clientWidth) popLeftAdjust=- popWidth-24
    else popLeftAdjust=0;
    if(MouseY+12+popHeight>document.body.clientHeight) popTopAdjust=- popHeight-24
    else popTopAdjust=0;
    dypopLayer.style.left=MouseX+12+document.body.scrollLeft+popLeftAdjust;
    dypopLayer.style.top=MouseY+12+document.body.scrollTop+popTopAdjust;
    dypopLayer.style.filter="Alpha(Opacity=0)";
    fadeOut();
}
//定义显示或隐藏提示层
function fadeOut()
{
    if(dypopLayer.filters.Alpha.opacity<popOpacity)
    {
        dypopLayer.filters.Alpha.opacity+=showPopStep;
    }
}

```

```
tFadeOut=setTimeout("fadeOut()",1);  
}  
else  
{  
    dypopLayer.filters.Alpha.opacity=popOpacity;  
    tFadeWaiting=setTimeout("fadeIn()",tPopShow);  
}  
}  
function fadeIn()  
{  
    if(dypopLayer.filters.Alpha.opacity>0)  
    {  
        dypopLayer.filters.Alpha.opacity -= 1;  
        tFadeIn=setTimeout("fadeIn()",1);  
    }  
}  
document.onmouseover=showPopupText;
```

需要在 body 中添加一个链接，用来调用并实现提示效果，代码如下所示，注意要指定链接的 title 属性，它是提示信息的来源。

```
<a href="http://www.google.com" title="我最喜爱的搜索">搜索</a>
```

【运行效果】

鼠标旁边的提示信息如图 4-13 所示。

【难点剖析】

本例中的重点是 div 层和 CSS 特效的应用。其实现原理如图 4-14 所示。

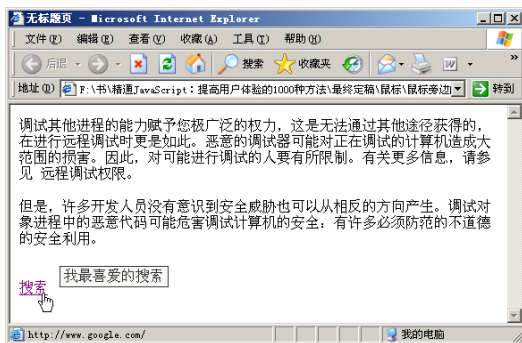


图 4-13 鼠标旁边的提示信息

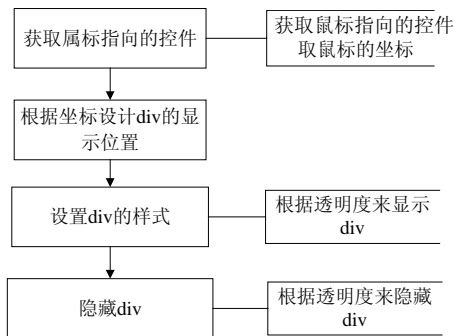


图 4-14 提示信息的实现原理

4.16 鼠标移到下拉框时自动全部打开

【实例描述】

用户要查看下拉框中的内容，需要单击下拉框或使用下拉框提供的三角箭头。本例将简化

这一操作，只要鼠标移动到下拉框处，就自动打开下拉框的列表。

【实现代码】

```
<html >
<head>
<title>标题页</title>
</head>
<body>
<select onmouseover="javascript:this.size=this.length" onmouseout="javascript:this.size=1">
<option>世界第一等</option>
<option>亚洲第一山</option>
<option>世界第一海</option>
</select>
</body>
</html>
```

【难点剖析】

本例的重点是下拉框的“onmouseover”事件和下拉框展开的方法。“onmouseover”事件当鼠标移动到该控件时被触发。展开下拉框使用的代码是“this.size=this.length”，其中“this.length”表示下拉框的所有选择项。

4.17 checkbox 鼠标移入移出的特效

【实例描述】

当鼠标移动到复选框时，复选框背景色会发生改变；当鼠标移走时，复选框恢复原来的颜色，这就是本例的特效。

【实现代码】

```
<html>
<script language="javascript">
var node;
var initied=false;
var init=function() //初始化方法
{
    node=document.getElementById("chk1"); //找到页面中的checkbox控件
    initied=true;
}
var check_over=function() //鼠标移入事件绑定的方法
{
    if(!initied)
        return;
    node.style.borderStyle="solid"; //设置边框
    node.style.borderColor="#FFCC00"; //设置边框颜色
    node.style.backgroundColor="#EEEEEE"; //设置背景色
}
var check_out=function() //鼠标移入事件绑定的方法
```



```
{
    node.style.borderStyle="none";           // 设置边框
    node.style.borderColor="#FFFFFF";        // 设置边框颜色为白色
    node.style.backgroundColor="#FFFFFF";    // 设置背景色
}
</script>
<body onload="init()">
<input type="checkbox" id="chk1" onmouseover="check_over()" onmouseout="check_out()" />
把鼠标移过来
</body>
</html>
```

【运行效果】

鼠标移入的效果如图 4-15 所示。

【难点剖析】

本例的重点是 checkbox 控件的“onmouseover”事件和“onmouseout”事件。“onmouseover”事件绑定的是“check_over”方法，此方法改变了 checkbox 的边框样式、边框颜色和背景色。“onmouseout”事件表示鼠标移走时的触发事件，其绑定了“check_out”方法，用来取消所有的特效设置。

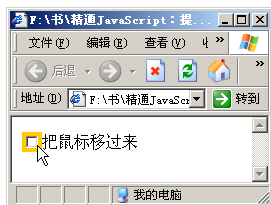


图 4-15 鼠标移入的效果

第 5 章

层和框架的特效

本章导读

目前，最流行的页面设计方法是 div+iframe。div 被称为层，是页面布局的常用控件。iframe 被称为框架，可以用来在当前页面中嵌入其他页面。本章介绍了常见的 div 和 iframe 应用技巧。



5.1 div 层提示效果

【实例描述】

当页面出现错误的时候，通常使用 alert 或者窗口给用户弹出错误信息，但这样会影响用户的操作。为了方便地显示错误信息，本例学习使用 div 层显示错误信息。

【实现代码】

```
<script language=javascript>
var div_x,div_y;
function div6()
{
    this.display=display;
}
//实现层和表格的动态创建
function display()
{
    document.write("<table align=center><tr><td><button style='width:100px; height:30px;font-size:12px;border:1px solid #A4B3C8;background-color:green;' type=button onclick=document.getElementById('div1').style.display='block' onfocus=this.blur()>div 留言</button></td></tr></table>");
    document.write("<div id='div1' style='font-size:12px;position:absolute; display:none;text-align:center;overflow:visible'>");
    document.write("<div style='position:absolute;top:expression((body. clientHeight-300)/2);left:expression((body.clientWidth-200)/2);width:200px;height:180px;background-color:#dbdbdb;border:1px solid #cccccc;'>");
    document.write("<table width=200 height=20 bgcolor=green onmousedown= 'div_x=event.x-parentNode.style.pixelLeft;div_y=event.y-parentNode.style.pixelTop;setCapture();' onmouseup='releaseCapture();' onmousemove='divMove(this. parentNode)' style='cursor:move;'>");
    document.write("<tr align=center>");
    document.write("<td align=left>提示:div6</td>");
    document.write("</tr>");
    document.write("</table>");
    document.write("<span style= cursor:hand onclick=this.parentNode. parentNode.style.display='none';><br>发生了严重的错误...<br>[ 返回]</span>");
    document.write(" </div>");
    document.write("</div>");
}
//实现层的拖移
function divMove(obj)
{
    if(event.button==1)
    {
        var divX=obj.clientLeft;
        var divY=obj.clientTop;
```

```

    obj.style.pixelLeft=divX+(event.x-div_x);
    obj.style.pixelTop=divY+(event.y-div_y);
  }
}
//创建并显示层
var mydiv=new div6();
mydiv.display();
</script>

```

【运行效果】

错误提示界面如图 5-1 所示,单击“返回”按钮,此提示会自动消失,还可以通过拖动提示消息的标题栏,实现 div 层的移动。

【难点剖析】

本例的重点是 div 层的动态创建和控制。使用“document.write”动态创建层,然后使用样式的“style.display”属性来控制层的隐藏或显示。当“style.display”的值为“none”时隐藏层,为“block”时显示层。



图 5-1 错误提示界面

5.2 层自动滚动到底端

【实例描述】

div 层的滚动条默认显示在顶端,有时候为了让用户可以一目了然地看到结果,可以将层的滚动条设置到最底端。本例学习这种效果。

【实现代码】

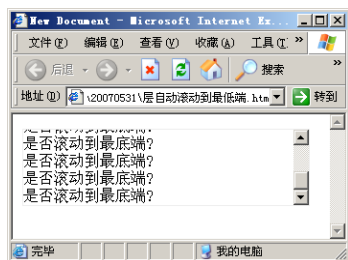
```

<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
<style type="text/css">
.test{
width:300px;
height:80px;
border:#eee solid 1px;
overflow-x:hidden;
overflow-y:scroll;
}
</style>
</HEAD>

```



```
<BODY onload="var t = document.getElementById('test'); t.scrollTop = t.scrollHeight;">
<div id="test" class="test">
是否滚动到底端?<br />
是否滚动到底端?<br />
是否滚动到底端?<br />
是否滚动到底端?<br />
是否滚动到底端?<br />
是否滚动到底端?<br />
是否滚动到底端?<br />
是否滚动到底端?<br />
是否滚动到底端?<br />
</div>
</BODY>
</HTML>
```



【运行效果】

本例的运行效果如图 5-2 所示。

【难点剖析】

本例的重点是 body 标签的“onload”事件。通过“getElementById”方法获取要设置滚动条所在的 div，然后设置 div 的“scrollTop”属性值为滚动条的高度。

图 5-2 本例的运行效果

5.3 div 的自动滚动

【实例描述】

div 可实现类似滚动公告栏的效果。本例学习如何使用 JavaScript，实现这种文本的滚动效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>如何实现 div 内自动滚动?</title>
<style type="text/css">
#scrollMsg{width:500px;height:235px;background-color:#feeff7;overflow:scroll;
overflow-x:hidden;text-overflow:ellipsis;word-break:break-all;}
#scrollMsg span{margin:6px;display:block;}
#scrollMsg span a{color:#f60;text-decoration:underline;margin:0 4px;}
#scrollMsg span a:hover{color:#f20;}
#scrollMsg span label{color:#c70060;margin:0 4px;}
</style>
<script type="text/javascript">
function getEid(id){
return document.getElementById(id); //获取指定的 div 元素
```



```

}
function newNode(param){
    return document.createElement(param);           //创建元素
}
function newTextNode(param){
    return document.createTextNode(param);           //创建元素内容
}

function scrollDiv(){
    var dest=getEid("scrollMsg");                     //获取要显示滚动内容的div
    var newStr=newTextNode(new Date().toLocaleString()+" :知识改变命运,科技推动发展!");
                                                    //显示的滚动信息
    var span=newNode("span");                         //创建span元素
    span.appendChild(newStr);                         //在span中添加显示信息
    dest.appendChild(span);                          //将span添加到div中
    scrollMsg.scrollTop+=10000;                       //滚动
    setTimeout("scrollDiv()",2000);                  //设置定时器定时滚动
}
window.onload=scrollDiv;
</script>
</head>
<body>
<div id="scrollMsg"></div>
</body>
</html>

```

【运行效果】

div 的自动滚动效果如图 5-3 所示。

【难点剖析】

本例的重点是动态创建元素。动态创建元素需要使用 JavaScript 的 DOM 对象,其可以实现元素的添加、删除、修改等功能。本例中,使用“createElement”方法创建了一个 span 元素,然后使用“createTextNode”方法为 span 元素指定文本内容,最后将 span 元素添加到要滚动的 div 中。

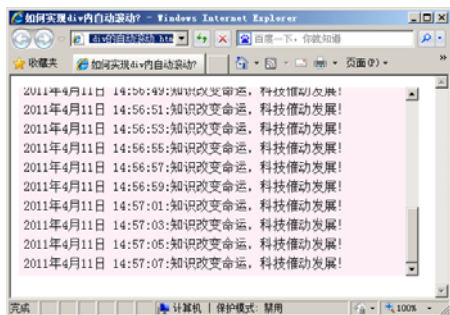


图 5-3 div 自动滚动的效果

5.4 div 的折叠效果

【实例描述】

折叠的 div 是网页显示内容的重要手段,因为使用折叠效果可以节省页面的空间,同时也提高了页面的美观性。本例学习如何制作一个简单的 div 折叠效果。

【实现代码】

```
<script type="text/javascript">
```



```
var mh = 30; //最小高度
var step = 1; //每次变化的px 量
var ms = 10; //每隔多久循环一次
//折叠速度的设置方法
function toggle(o){
    if (!o.tid)o.tid = "_" + Math.random() * 100;
    if (!window.toggler)window.toggler = {};
    if (!window.toggler[o.tid]){
        window.toggler[o.tid]={
            obj:o,
            maxHeight:o.offsetHeight,
            minHeight:mh,
            timer:null,
            action:1
        }; }
    o.style.height = o.offsetHeight + "px";
    if (window.toggler[o.tid].timer)clearTimeout(window.toggler[o.tid].timer);
    window.toggler[o.tid].action *= -1;
    window.toggler[o.tid].timer = setTimeout("anim('"+o.tid+"')",ms );//注意计时器的用法
}
//通过对象的最小高度和最大高度，判断折叠是否停止
function anim(id){
    var t = window.toggler[id];
    var o = window.toggler[id].obj;
    if (t.action < 0){
        if (o.offsetHeight <= t.minHeight){
            clearTimeout(t.timer);
            return;
        }
    }
    else{
        if (o.offsetHeight >= t.maxHeight){
            clearTimeout(t.timer);
            return;
        }
    }
    o.style.height = (parseInt(o.style.height, 10) + t.action * step) + "px";
    window.toggler[id].timer = setTimeout("anim('"+id+"')",ms );
}
</script>
```

【运行效果】

div 的折叠效果如图 5-4 所示，展开效果如图 5-5 所示。

【难点剖析】

本例的重点是对象的创建，如“toggler”就是自己创建的一个对象，其包含 obj、maxHeight 等属性，可以在脚本方法中调用这些属性，此实现类似于面向对象的设计方法。

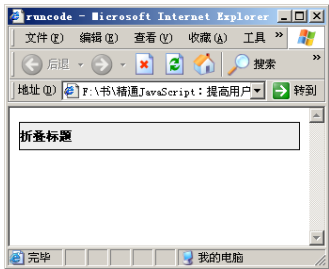


图 5-4 div 折叠效果

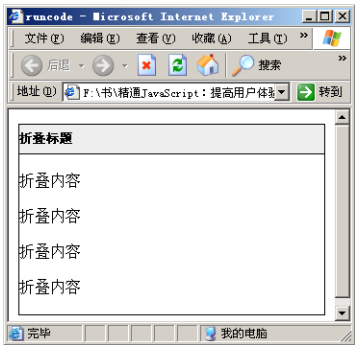


图 5-5 div 展开效果

5.5 圆角 div

【实例描述】

为了增加页面的美观效果，有时需要将 div 层或表格的边角进行圆角处理。本例学习如何实现 div 层的圆角效果，这一功能也属于目前比较流行的 Ajax 技术。

【实现代码】

```
<html xmlns:v>
<head>
<style>
    v\:*{behavior: url(#default#VML);}
</style>
</head>
<body>
    <v:roundRect style="position:absolute;left:20px;top:50px;width:200px; height:
140px;" FillColor="#AAEAFa" Filled="T" />
</body>
</html>
```

【运行效果】

圆角 div 的运行效果如图 5-6 所示。

【难点剖析】

本例的重点是使用了 VML 技术。VML 的全称是 Vector Markup Language (矢量可标记语言)，是 IE 里面的画笔工具，可以用其画出所要的图形，而且结合脚本可让图形产生动态的效果。VML 是微软于 1999 年 9 月随 IE 5.0 一起发布的，也就是说在 IE 5.0 以上都能使用 VML。注意使用 VML 时，需要调用特殊的命名空间，如“<html xmlns:v>”。



图 5-6 圆角 div 的运行效果



5.6 动态添加 iframe 框架

【实例描述】

iframe 框架用来加载页面。本例将学习如何动态添加 iframe 到页面中。

【实现代码】

```
<html>
<head>
<title> 动态添加框架 </title>
<script language="JavaScript">
window.onload = function()
{
var iframe = document.createElement('iframe');           //动态创建框架
iframe.src="http://www.google.com";                       //框架中加载的页面
document.body.appendChild(iframe);                       //将框架添加到当前窗体
}
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是 DOM 方法的运用。“createElement”方法用来在页面中创建元素，“appendChild”方法用来将元素添加到网页中。

5.7 用层实现长篇文章分页

【实例描述】

在很多读书或新闻网站中，为了提高可读性，通常会对长篇文章进行分页。本例将学习如何对长篇文章分页。

【实现代码】

```
<html>
<head>
<style type="text/css">
#frameContent{
width:500px;           /*调整显示区的宽*/
height:200px;         /*调整显示区的高*/
font-size:14px;
line-height:20px;
border:1px solid #000000;
overflow-pageINdex:hidden;
```

```

        overflow-y:hidden;
        word-break:break-all;
    }
    a{
        font-size:12px;
        color:#000000;
        text-decoration:underline;
    }
    a:hover{
        font-size:12px;
        color:#CC0000;
        text-decoration:underline;
    }
</style>
</head>
<body>
<div id="frameContent">
    在这里输入你的文章
</div>
<P>
<div id="pages" style="font-size:12px;"></div>
<script language="javascript">
var obj = document.getElementById("frameContent"); //获取内容层
var pages = document.getElementById("pages"); //获取翻页层
window.onload = function() //重写窗体加载的事件
{
    var allpages = Math.ceil(parseInt(obj.scrollHeight)/parseInt(obj.offsetHeight));//
    获取页面数量
    pages.innerHTML = "<b>共"+allpages+"页</b>"; //输出页面数量
    for (var i=1;i<=allpages;i++){
        pages.innerHTML += "<a href=\"javascript:showPage('"+i+"')\">第"+i+"页
</a>&nbsp;";
        //循环输出第几页
    }
}
function showPage(pageIndex)
{
    obj.scrollTop=(pageIndex-1)*parseInt(obj.offsetHeight);
    //根据高度，输出指定的页
}
</script>
</body>
</html>

```

【运行效果】

长篇文章的分页效果如图 5-7 所示。

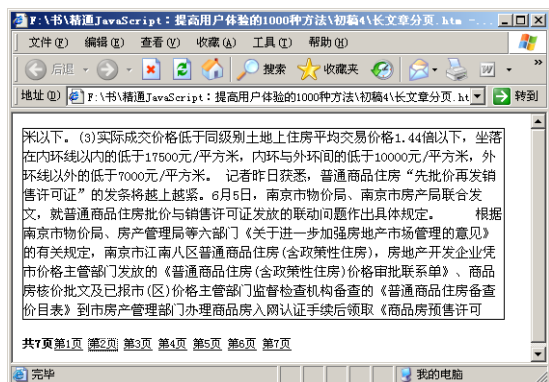


图 5-7 长篇文章的分页效果

【难点剖析】

本例将所有文章放在一个 div 层中，然后使用 CSS 样式表固定这个 div 的高度为 200px。如果文章的内容超过了 200px，则 div 会出现滚动条，此时使用“scrollHeight”获得垂直滚动条的高度，用此值除以 div 层的高度，取整后就是总的页数。当用户查看某页时，便调用“showPage”方法，实现翻页效果。

5.8 iframe 自适应高度

【实例描述】

iframe 框架可以动态加载网页，并能嵌入到当前页面中，实现页面嵌套的功能。为了更好地显示子页，本例学习如何自动调整 iframe 的高度。

【实现代码】

本例有两个页面，第一个页面“iframe 自适应高度 1.htm”中有一个 iframe，实现代码如下所示：

```
<html>
<head>
<title>iframe 自适应高度</title>
</head>
<body>
<div><iframe src="iframe 自适应高度.htm"></iframe></div>
</body>
</html>
```

第二个页面会自动调整子页的大小，实现代码如下所示：

```
<html>
<head>
<title>iframe 自适应网页</title>
<script type="text/javascript">
```

```

//自动调整 iframe 框架的方法
function iframeAuto()
{
try
{
if(window!=parent)
{
//定位需要调整的 frame 框架（在父级窗口中查找）
var a = parent.document.getElementsByTagName("IFRAME");
for(var i=0; i<a.length; i++)
{
if(a[i].contentWindow==window)
{
var h1=0, h2=0;
a[i].parentNode.style.height = a[i].offsetHeight + "px";
a[i].style.height = "10px";           //首先设置高度为 10px,后面会修改
if(document.documentElement&&document.documentElement.scrollHeight)
{
h1=document.documentElement.scrollHeight;
}
if(document.body) h2=document.body.scrollHeight;
var h=Math.max(h1, h2);           //取两者中的较大值
if(document.all) {h += 4;}
if(window.opera) {h += 1;}
//调整框架的大小
a[i].style.height = a[i].parentNode.style.height = h + "px";
} } }
}
catch (ex){}
}
//事件绑定的方法，支持 IE5 以上版本
if(window.attachEvent)
{
window.attachEvent("onload", iframeAuto);
}
else if(window.addEventListener)
{
window.addEventListener('load', iframeAuto, false);
}
//-->
</script>
</head>
<body>
<table border="1" width="200" style="height: 400px; background-color: gray">
<tr>
<td>iframe 自适应其加载的网页</td>
</tr>
</table>

```



</body>

</html>

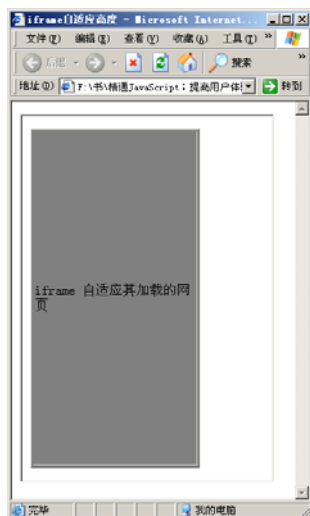


图 5-8 iframe 自适应高度实例的运行效果【难点剖析】

【运行效果】

iframe 自适应高度实例的运行效果如图 5-8 所示。

本例的重点有如何获取父级页面中的 iframe、如何实现 iframe 的自动调整。获取父级页面中的 iframe，主要依靠“parent.document.getElementsByTagName("IFRAME");”语句，其中的“parent”便是获取的父级窗口。自动调整功能实现是将调整方法绑定到 onload 事件中，即窗体一加载便调整 iframe 的大小。

5.9 类似 MSN 的消息提示

【实例描述】

当 MSN 中有邮件或消息时，会在右下角出现一个提示窗口。本例学习如何制作此类型的提示效果。

【实现代码】

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>类 MSN 提示的页面效果</title>
<script language="JavaScript">
window.onload = viewMsg;           //加载页面时，即刻获取短消息
window.onresize = resizeDiv;       //根据窗体高度和宽度，改变短消息提示框的高度和宽度
window.onerror = function(){}      //出现错误时，不做任何处理
var divTop,divLeft,divWidth,divHeight,docHeight,docWidth,objTimer,i = 0;
                                   //关于位置的相关变量

function viewMsg()
{
    try
    {
        divTop = parseInt(document.getElementById("divMsg").style.top,10) //div 的 x 坐标
        divLeft = parseInt(document.getElementById("divMsg").style.left,10) //div 的 y 坐标
        divHeight = parseInt(document.getElementById("divMsg").offsetHeight,10)
                                                //div 的高度
        divWidth = parseInt(document.getElementById("divMsg").offsetWidth,10) //div 的宽度
        docWidth = document.body.clientWidth; //窗体宽度
        docHeight = document.body.clientHeight; //窗体高度
        document.getElementById("divMsg").style.top = parseInt(document.body.scrollTop,10)
+ docHeight + 10; //设置 div 的 y 坐标
        document.getElementById("divMsg").style.left = parseInt(document.body.
scrollTop,10) + docWidth - divWidth //设置 div 的 x 坐标
```



```

        document.getElementById("divMsg").style.visibility="visible" //设置div 显示
        objTimer = window.setInterval("moveDiv()",10) //设置定时器
    }
    catch(e){}
}
function resizeDiv()
{
    i+=1
    if (i>500) closeDiv()
    try
    {
        divHeight = parseInt(document.getElementById("divMsg").offsetHeight, 10) //设置div 高度
        divWidth = parseInt(document.getElementById("divMsg").offsetWidth,10) //设置div 宽度
        docWidth = document.body.clientWidth; //获取窗体宽度
        docHeight = document.body.clientHeight; //设置窗体高度
        document.getElementById("divMsg").style.top = docHeight - divHeight + parseInt
(document.body.scrollTop,10) //设置div 的y 坐标
        document.getElementById("divMsg").style.left = docWidth - divWidth + parseInt
(document.body.scrollLeft,10) //设置div 的x 坐标
    }
    catch(e){}
}
function moveDiv()
{
    try
    {
        if (parseInt(document.getElementById("divMsg").style.top,10) <= (docHeight -
divHeight + parseInt(document.body.scrollTop,10)))
        {
            window.clearInterval(objTimer)
            objTimer = window.setInterval("resizeDiv()",1) //调整div 的位置和大小
        }
        divTop = parseInt(document.getElementById("divMsg").style.top,10) //获取y 坐标
        document.getElementById("divMsg").style.top = divTop - 1 //调整div 的y 坐标
    }
    catch(e){}
}
function closeDiv()
{
    document.getElementById('divMsg').style.visibility='hidden'; //将短信息提示层隐藏
    if(objTimer) window.clearInterval(objTimer); //清除定时器
}
</script>
</head>

```

```
<body scroll="no">
注意右下角的提示<br />

<DIV id=divMsg style="BORDER-RIGHT: #455690 1px solid; BORDER-TOP: #a6b4cf 1px solid;
Z-INDEX:99999; LEFT: 0px; VISIBILITY: hidden; BORDER-LEFT: #a6b4cf 1px solid; WIDTH: 180px;
BORDER-BOTTOM: #455690 1px solid; POSITION: absolute; TOP: 0px; HEIGHT: 116px; BACKGROUND-COLOR:
#c9d3f3">

    <TABLE style="BORDER-TOP: #ffffff 1px solid; BORDER-LEFT: #ffffff 1px solid"
cellSpacing=0 cellPadding=0 width="100%" bgColor=#cfdef4 border=0>

        <TBODY>

            <TR>

                <TD style="FONT-SIZE: 12px;COLOR: #0f2c8c" width=30 height=24></TD>
                <TD style="FONT-WEIGHT: normal; FONT-SIZE: 12px;COLOR: #1f336b; PADDING-TOP:
4px;PADDING-left: 4px" vAlign=center width="100%"> 短消息提示: </TD>

                <TD style="PADDING-TOP: 2px;PADDING-right:2px" vAlign=center align=right
width=19><span title=关闭 style="CURSOR: hand;color:red;font-size: 12px;font-weight:
bold;margin-right:4px;" onclick=closeDiv() >X</span></TD>

            </TR>

            <TR>

                <TD style="PADDING-RIGHT: 1px;PADDING-BOTTOM: 1px" colSpan=3 height=90>
                <DIV style="BORDER-RIGHT: #b9c9ef 1px solid; PADDING-RIGHT: 13px;
BORDER-TOP: #728eb8 1px solid; PADDING-LEFT: 13px; FONT-SIZE: 12px; PADDING-BOTTOM: 13px;
BORDER-LEFT: #728eb8 1px solid; WIDTH: 100%; COLOR: #1f336b; PADDING-TOP: 18px; BORDER-BOTTOM:
#b9c9ef 1px solid; HEIGHT: 100%">您有<font color=#FF0000>1</font>条新消息<BR><BR>

                <DIV align=center style="word-break:break-all"><a href=
"Javascript:alert('内容:好久不见,出来吃饭吧')"><font color=#FF0000>点击查看短信</font></a></DIV>

                </DIV>

            </TD>

        </TR>

    </TBODY>

</TABLE>

</DIV>

</body>

</html>
```

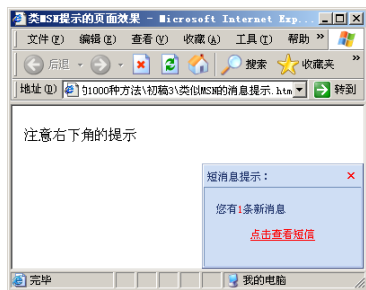


图 5-9 类 MSN 的提示效果

【运行效果】

类 MSN 的提示效果如图 5-9 所示。

【难点剖析】

本例中定义了 div 消息层的 4 个常用方法：

- “viewMsg”方法用来显示提示信息，显示时根据窗体的高度和宽度，调整 div 层的位置和大小；
- “resizeDiv”方法用来调整 div 层的大小；
- “moveDiv”方法用来设置 div 层的 y 坐标，一般用于窗体大小和位置改变时；

- “closeDiv”方法用来隐藏 div，调用 div 的“visibility”属性，并设置其值为“hidden”。

5.10 只打印 iframe 的内容

【实例描述】

在打印页面时，可以通过正则或某个设计标识，只打印页面的部分内容。那么该如何只打印嵌入到此页面中框架的内容呢？本例学习一个只打印部分内容的新方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<body>
<input onclick='prn()' type=button value=print_Ifram5.10e><br>
<iframe id=myframe src="http://www.baidu.com"></iframe>
<script>
function prn()
{
var win=window.open("about:blank")           //打开一个空页面
win.moveTo(100,100)                           //移动到指定位置
win.location=document.all.myframe.src         //指定页面的内容
win.print()                                    //打印页面
}
</script>
</body>
</html>
```

【难点剖析】

当用户单击“print_Iframe”按钮时，其实在后台会打开一个新页面，此页面的内容其实就是框架中的全部内容。然后在新页面中，调用“print”方法实现框架内容的打印，其实就是一个新页面的打印。

第 6 章

下拉列表特效

本章导读

下拉列表用来封装一些类型相同的数据，在 HTML 中用 select 标签，属于数据分类控件。本章介绍 select 在 Web 开发中常见的一些应用技巧，如动态添加下拉列表中的项、遍历下拉列表等。

6.1 下拉列表框实现多选

【实例描述】

下拉列表框用来实现多个相同类型数据的显示，用户有时需要在这些数据中选择多个。本例学习如何实现下拉列表框的多选。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<select name="sel" multiple>
<option> 首都图书馆</option>
<option> 首都新华书店</option>
<option> 首都数字图书馆</option>
<option> 中关村图书大厦</option>
<option> 爱书网图书在线</option>
</select>
</body>
</html>
```

【运行效果】

下拉列表框多选效果如图 6-1 所示。

【难点剖析】

本例的重点是下拉列表框的“multiple”属性。此属性控制下拉列表框是否允许多选。如果设置了此属性，则通过“Ctrl+鼠标左键”可实现选项的多选。



图 6-1 下拉列表框多选效果

6.2 实现两个 select 的同步

【实例描述】

当选定第一个下拉列表框的时候，第二个下拉列表框的值也随之改变，这被称为两个下拉列表框的同步。本例学习如何实现这种同步效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
```

[illegible]

图 6-2 选中第一个下拉列表框后的效果

【运行效果】

选中第一个下拉列表框后的效果如图 6-2 所示。

【难点剖析】

本例的技巧就是 `select` 标签的“`selectedIndex`”属性和“`onchange`”事件。当用户选择第一个下拉列表框后，第二个下拉列表框也要改变，所以要将此改变添加到第一个下拉列表框的“`onchange`”事件中。“`selectedIndex`”属性用来获取当前 `select`

标签的选项索引,当知道第一个下拉列表框的选项索引后,使用“options[selectedIndex].selected”就可以自动设置第二个下拉列表框的选择项。

6.3 被选中的列表项下次不能再选

【实例描述】

本例中列表项的选择类似于单选按钮，用户只能选择一个，而且选择后，列表项不能再改变。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function selChange(sel)                                // 选择后触发的事件
{
    sel.disabled = true;                                // 某项不能使用
}
```

```

</script>
</head>
<body>
<select onchange="selChange(this)">
  <option>--请选择--</option>
  <option>第一项</option>
  <option>第二项</option>
  <option>第三项</option>
</select>
</body>
</html>

```

【难点剖析】

本例主要是“onchange”事件，用来捕获当列表项发生改变时的事件。使用“disabled”属性来控制下拉列表框为不可选状态。

6.4 不带滚动条的 select

【实例描述】

默认的 select 控件，是通过下拉的形式提供选项，有时候为了页面布局，需要使用其中的滚动条。本例学习如何去掉 select 的滚动条，并显示 select 的所有选项。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<select size="2" style="width:150px" name="myse1" style="overflow:hidden" multiple>
  <option>aaa</option>
  <option>bbb</option>
  <option>ccc</option>
  <option>ddd</option>
  <option>eee</option>
  <option>fff</option>
</select>
<script language="javascript">
document.all.myse1.size=document.all.myse1.options.length; //根据列表框内容改变列表框的高度
</script>
</body>
</html>

```

【运行效果】

不带滚动条的 select 如图 6-3 所示。



图 6-3 不带滚动条的 select

【难点剖析】

本例，首先默认设置 select 的高度只能容纳两个选项。当 select 初始化并加载数据完成后，再使用“document.all.myself.options.length”获取 select 中选项的个数。最后设置“document.all.myself.size”为实际的选项个数。

6.5 从一个下拉列表往另一个下拉列表添加内容

【实例描述】

在网页中经常需要用户选择一些类似的信息（如个人爱好），为了明确用户的选择，通常用列表列出几乎所有的爱好，然后用户选择自己的爱好内容添加到另外一个列表中。本例就是实现这种选择效果。

【实现代码】

```
<script language="JavaScript">
//设计字符对象的 trimEnd 方法
String.prototype.trimEnd = function(trimString) {
    var re = new RegExp(trimString+"*$", "g");
    return this.replace(re, "");
};
//设计数组对象的 indexOf 方法
Array.prototype.indexOf = function(itemValue) {
    var nIndex = -1;
    for (var i=0; i<this.length; i++)
    {
        if (this[i] == itemValue) nIndex = i;
    }
    return nIndex;
};
//定义变量，变量为当前文档中的控件对象
var oSrc = document.getElementById("oldSelect");
var oTarget = document.getElementById("newSelect");
var oCopy = document.getElementById("btnMove");
//定义按钮的 click 事件
oCopy.onclick = function() {
    var aSelectedIndexes = getSelectedIndexes(oSrc);
    for (var i=0; i<aSelectedIndexes.length; i++)
    {
        var oOption = document.createElement("OPTION");
        oOption.text = oSrc.options[aSelectedIndexes[i]].text;
        oOption.value = oSrc.options[aSelectedIndexes[i]].value;
        oTarget.options.add(oOption);
    }
}
```



```

    }
};
//获取源下拉列表中被选择的内容
function getSelectedIndexes(oSrc)
{
    var aSelectedIndexes = new Array();

    for (var i=0; i<oSrc.options.length; i++)
    {
        if (oSrc.options[i].selected)
            aSelectedIndexes[aSelectedIndexes.length] = i;
    }

    return aSelectedIndexes;
}
//单击源下拉列表的事件
oSrc.onclick = function() {
    this.selectedIndexes = getSelectedIndexes(this);
    if (this.selectedIndexes.length == 1)
        this.options[this.selectedIndexes].selected = false;
};
//改变源下拉列表选择的事件
oSrc.onchange = function() {
    var j = this.selectedIndexes.indexOf(this.selectedIndex);
    if (j > -1)
    {
        this.options[this.selectedIndex].selected = false;
        this.selectedIndexes.splice(j, 1);
    }
    if (this.selectedIndexes.length > 0)
    {
        var nSelectedIndex;
        for (var i=0; i<this.selectedIndexes.length; i++)
        {
            nSelectedIndex = parseInt(this.selectedIndexes[i]);
            this.options[nSelectedIndex].selected = true;
        }
    }
};
</script>

```

【运行效果】

列表选择的效果如图 6-4 所示。

【难点剖析】

本例的难点是如何判断下拉列表的选择内容，还有如何动态添加到另外一个下拉列表中。判断选择列表的内容需要通过列表索引获得，本例使用了方法“getSelectedIndexes”。动态在下



图 6-4 列表选择的效果

在列表中创建元素,使用了 DOM 的方法“createElement”来创建“option”元素,最后动态地为此元素赋值。

6.6 改变列表项的上下顺序

【实例描述】

列表项的上下移动,是网页中经常使用的特效。本例通过一个简单的例子,学习这种特效的制作。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<select size="10" name="select">
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
</select>
<input type="button" value="向上移动" onclick="doUp()">
<input type="button" value="向下移动" onclick="doDown()">
<script>
function doOrder(select,sequence)                                //将上、下两个方法合并成一个
{
    if (!select||select.selectedIndex==0)                        //如果没有选择列表项,不进行任何操作
        return false;
    with (select)
    {
        var newIndex = selectedIndex + sequence;                //获取移动后的索引
        var oldIndex = selectedIndex;                            //旧索引
        if (newIndex>=options.length||newIndex<0||sequence==0||newIndex<0)
                                                                //判断是否超出边界
        {
            return false;
        }
        options[newIndex].swapNode(options[oldIndex]) //交换指定索引处的节点
    }
}
```

```

        return true;
    }
    function doUp()
    {
        doOrder(document.all.select,-1);           // 向上移动的方法
    }
    function doDown()
    {
        doOrder(document.all.select,1);           // 向下移动的方法
    }
</script>
</body>
</html>

```

【运行效果】

本例的运行效果如图 6-5 所示。

【难点剖析】

本例的重点是要移动的元素的位置和新位置，旧位置使用“selected Index”属性获取，新位置根据选择的是“向上移动”还是“向下移动”，下就“+1”，上就“-1”。最后使用“swapNode”方法实现新旧位置的互换。



图 6-5 本例的运行效果

6.7 给下拉列表框数据分组

【实例描述】

如果下拉列表框的内容不相近，可以用多个下拉列表框表示，如部门列表、人员列表。如果下拉列表框的内容比较相近，此时可以通过一个下拉列表框实现，并在其中设置不同的组。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<body>
<SELECT>
<OPTGROUP LABEL="山东省">
<OPTION>济南</OPTION>
<OPTION>青岛</OPTION>
<OPTION>潍坊</OPTION>
</OPTGROUP>
<OPTGROUP LABEL="河北省">
<OPTION>保定</OPTION>
<OPTION>沧州</OPTION>
<OPTION>石家庄</OPTION>
</OPTGROUP>

```

```
</body>  
</html>
```

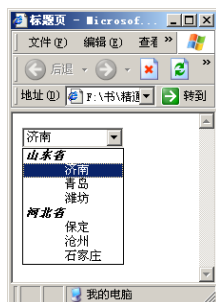


图 6-6 数据分组后的下拉列表框

【运行效果】

数据分组后的下拉列表框如图 6-6 所示。

【难点剖析】

本例的重点是下拉列表框中的“组”。“option”表示下拉列表框中的某项，而“optgroup”则表示下拉列表框中的某组。

6.8 获取下拉列表框的选择

【实例描述】

下拉列表框在网页中由 select 组成，其中包括多个 option 选择。本例学习如何获取用户选择的项。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
<script LANGUAGE="JavaScript">  
function mySelect()  
{  
    var mysel=document.getElementById("select1");    //获取文档中的下拉列表框  
    var txt=mysel.options[mysel.options.selectedIndex].text;    //获取下拉列表框显示的文本  
    var valu=mysel.options[mysel.options.selectedIndex].value;    //获取下拉列表框的值  
    alert("您选择的是"+ valu);  
}  
</script>  
</head>  
<body>  
<select size="10" ondblclick="mySelect()" id="select1">  
<option value="大中城市">大城市</option>  
<option value="国际化大都市">国际城市</option>  
</select>  
</body>  
</html>
```

【运行效果】

双击选项后的效果如图 6-7 所示。

【难点剖析】

列表框包括两个数据：值和文本。值（value）是选择项的唯一标识，并不显示给用户。文本（text）显示给用户，不需要唯一。

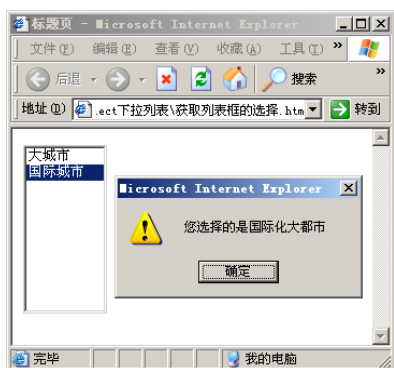


图 6-7 双击选项后的效果

6.9 类 IE 下拉列表框

【实例描述】

IE 下拉列表框除实现数据列表选择外，还可以实现页面导航功能。本例使用 `select` 和 `input` 实现一个此类型的控件。

【实现代码】

```
<html>
<head>
<script language="javascript">
    function selectChange()
    {
        document.all.text.value = document.all.select.value;
        window.location.href= document.all.text.value;
    }
</script>
</head>
<body>
<div>
<select id='select' style="position:absolute;width:300px;clip: rect(0 320 260 280)"
onchange="selectChange()">
<option value='http://www.google.com'>"http://www.google.com</option>
<option value='>http://www.baidu.com'>http://www.baidu.com </option>
<option value='http://www.sina.com.cn ' > http://www.sina.com.cn </option>
<option value='http://www.sohu.com'> http://www.sohu.com </option>
</select>
<input type='text' id='text' style="position:absolute;width:300px;" value=
"http://www.google.com" onfocus="this.value='';document.all.select.value='';">
</div>
</body>
</html>
```



图 6-8 类 IE 下拉列表框控件的效果

【运行效果】

类 IE 下拉列表框控件的效果如图 6-8 所示。

【难点剖析】

本例的实现原理是首先判断下拉列表框的选择事件，并获取用户的选择值，然后将用户的选择值显示在 input 控件内，最后使用“window.location.href”实现选择值的导航。

6.10 下拉列表框式邮件发送

【实例描述】

通过下拉列表框方式选择一个收信人，然后单击“发信”按钮，调用本机的 Outlook，实现发信的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<form name="addresses">
请选择给谁发信<select name="list" onChange="update()">
<option value="mailto:admin@google.com">给 Google 管理员写信
<option value="mailto:support@263.com">给 263 客服写信
<option value="mailto:admin@263.net">给 263 管理员写信
</select>
<a href="updatelink">发信!</a>
</form>
<script language="javascript">
    pos = 666;
    for(num=0;num<document.links.length;num++) {
        if (document.links[num].href.indexOf("updatelink") != -1) {
            //是否已经指定邮件地址
            pos = num;
            //更改 pos 变量
            num = 300;
        }
    }
    function update() {
        if (pos!=666) {
            sel = document.addresses.list.selectedIndex;
            //判断选择的邮件
            document.links[pos].href = document.addresses.list[sel].value;
            //登记邮件地址
        }
    }
}
```

```

update();
</script>
</body>
</html>

```

【运行效果】

单击单元格后的效果如图 6-9 所示。

【难点剖析】

本例的重点是如何将选择的收信人添加到窗体的地址列表。使用“selectedIndex”判断用户选择的收信人，然后通过“a”标签调用操作系统的 Outlook，实现发信功能。



图 6-9 单击单元格后的效果

6.11 手动调整的列表框

【实例描述】

列表框用来显示一组相似的信息，本例学习如何动态实现列表框内容的添加、删除、上移和下移。

【实现代码】

```

<script language="javascript">
function move(listItem,listObj) {                                //列表项添加的方法
    var i = 0;
    if(listItem.value != "") {                                    //如果列表项不为空
        var newOp = new Option();                                //创建一个列表项
        newOp.value = listItem.value;                            //设置列表项的值
        newOp.text = listItem.value;                             //设置列表项的文本
        listObj.options[listObj.options.length] = newOp;        //添加新项到列表框中
        listItem.value = "";
    }
}

function remove(listObj) {                                        //列表框的删除方法
    for(var i=0; i<listObj.options.length; i++) {
        if(listObj.options[i].selected && listObj.options[i] != "") {
            listObj.options[i].value = "";                        //清除选定项的值
            listObj.options[i].text = "";                         //清除选定项的文本
        }
    }
    delAfter(listObj);                                           //删除后的排序处理
}

function delAfter(alistObj) {
    for(var i = 0; i < alistObj.options.length; i++) {
        if(alistObj.options[i].value == "") {                    //值为空的会被清除

```



```

        for(var j = i; j < alistObj.options.length - 1; j++) {
            alistObj.options[j].value = alistObj.options[j + 1].value;
            //重新排列顺序
            alistObj.options[j].text = alistObj.options[j + 1].text;
        }
        var ln = i;
        break;
    }
}
if(ln < alistObj.options.length) {
    alistObj.options.length -= 1;
    delAfter(alistObj);
}
}
function Moveup(listObj) { //将列表项往上移动
    for(var i = 0; i < listObj.options.length; i++) { //遍历列表项
        if (listObj.options[i].selected && listObj.options[i] != "" && listObj.
options[i] != listObj.options[0]) {
            var tmpvall1 = listObj.options[i].value; //获取当前项的值
            var tmpvall2 = listObj.options[i].text; //获取当前项的文本
            listObj.options[i].value = listObj.options[i - 1].value; //获取上一项的值
            listObj.options[i].text = listObj.options[i - 1].text //获取上一项的文本
            listObj.options[i-1].value = tmpvall1; //实现上下值的互换
            listObj.options[i-1].text = tmpvall2; //实现上下文本的互换
        }
    }
}
function Movedown(listObj) { //将列表项往下移动
    for(var i = 0; i < listObj.options.length; i++) {
        if (listObj.options[i].selected && listObj.options[i] != "" && listObj.
options[i+1] != listObj.options[listObj.options.length]) {
            var tmpvall1 = listObj.options[i].value; //获取当前项的值
            var tmpvall2 = listObj.options[i].text; //获取当前项的文本
            listObj.options[i].value = listObj.options[i+1].value; //获取下一项的值
            listObj.options[i].text = listObj.options[i+1].text //获取下一项的文本
            listObj.options[i+1].value = tmpvall1; //实现上下值的互换
            listObj.options[i+1].text = tmpvall2; //实现上下文本的互换
        }
    }
}
}
</script>

```

【运行效果】

本例运行的效果如图 6-10 所示。

【难点剖析】

本例的重点是对列表框的遍历。当添加新项到列表框时，通过创建新 option 元素的方式添

加项。当删除选项时，先设置选项的值为空，然后以遍历的方式将值为空的项删除。上移和下移的方法都是通过遍历列表框，然后进行上下选项值的互换。

6.12 下拉框式网站导航

【实例描述】

本例的导航并不是菜单，而是对网站中各个模块的描述。用户可以通过下拉列表框，了解网站中各个模块的功能。

【实现代码】



图 6-10 本例运行的效果

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var messages = new Array(6);                                //显示的信息数组
messages[0] = "";
messages[1] = "这里查询所有的健康咨询，学习如何保护自己的身体，如何正确饮食！";
messages[2] = "是刚注册的用户吗？还不知道如何浏览本站吧，从这里进入即可！";
messages[3] = "要下载常用软件吗/本站的某些页面打不开吗?从这里进去看看！";
messages[4] = "不管你有什么想说的,都可以来这里发泄！";
messages[5] = "这里有所有资深会员的经验之谈,进来看看吧!";
function messageReveal()
{
    var messageindex = document.messageForm.messagePick.selectedIndex //获取用户选择的列表索引
    document.messageForm.messageField.value = messages[messageindex]; //显示指定的信息
}
</SCRIPT>
</head>
<body>
<form name="messageForm">
<select name="messagePick" OnChange="messageReveal()">
<option value="0">网站导航条
<option>健康杂谈
<option>新用户帮助
<option>下载软件
<option>灌水论坛
<option>老手经验
</select>
<br><p>
<textarea name="messageField" rows=6 cols=50 wrap=virtual></textarea>
</form>
</body>
```

</html>

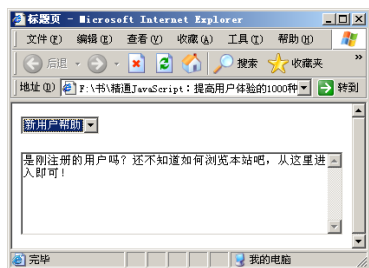


图 6-11 本例的运行效果

【运行效果】

本例的运行效果如图 6-11 所示。

【难点剖析】

本例的重点是 select 标签的应用。select 使用下拉框显示一组数据，通过索引访问数据中的元素。其“selectedIndex”属性用来表示用户选中的索引，本例通过该属性获取数组中指定的提示信息。

6.13 综合的搜索引擎

【实例描述】

搜索是获取网络信息的重要方法。本例通过一个简单的页面，学习如何整合多个搜索引擎为自己所用。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<form Name="InputForm">
<div align="center"><center><p>
<script language="JavaScript">
var FirstForm;
function StartSearch()
{
//使用隐藏控件保存用户输入的查询参数
document.forms[FirstForm+document.InputForm.SearchSelect.
selectedIndex].elements[0].value=document.InputForm.SearchWords.value;
//提交查询参数到指定网站
document.forms[FirstForm+document.InputForm.SearchSelect.selectedIndex].submit();
}
</script>
<span style="font-size: 9pt">查找内容:</span>
<input name="SearchWords" type="text" size="21" style=" margin-left: 1px"><br>
<span style="font-size: 9pt">搜索引擎:</span>
<select Name="SearchSelect" size="1" >
<option selected>英文 Yahoo</option>
<option value="Google 搜索">中文 Google</option>
<option value="百度搜索">百度中文搜索</option>
</select><br>
<input type="button" value=" 开始查找 " onClick="StartSearch()">
```

```
<script language="JavaScript">
FirstForm=document.forms.length
</script></p>
</center></div>
</form>
<form action="http://search.yahoo.com/bin/search" method="get">
<input type="hidden" name="p" value>
</form>
<form action="http://www.google.com/search">
<input type="hidden" name="q" value>
</form>
<form action="http://www.baidu.com/s">
<input type="hidden" name="wd" value>
</form>
</body>
</html>
```

【运行效果】

整合的搜索引擎的运行效果如图 6-12 所示。打开的搜索结果页如图 6-13 所示，注意其地址栏的搜索参数，如果搜索内容为中文，则会被编码化。



图 6-12 整合的搜索引擎的运行效果



图 6-13 搜索结果页

【难点剖析】

本例的重点是在页面中添加多个 form，然后使用 form 的“action”方法提交当前页面到搜索页面。在提交时要注意修改搜索参数，因为不同的搜索引擎，其搜索参数不同。如本例中 google 的搜索参数为“q”，百度的搜索参数为“wd”。

6.14 经典的 ListView 列表框

【实例描述】

ListView 是 C/S 开发中的一个控件，类似于资源管理器中浏览文件的效果。本例使用 Microsoft 提供的一个 Object 组件，实现 ListView 浏览效果。



【实现代码】

```
<script language = 'javascript'>
function InitList(theList, theTable, iEnd, iId, checkIt) //初始化列表的方法
{
    var colWidth = (document.body.clientWidth - 200) / iEnd //设置列宽度
    with(theList)
    {
        View = 3 //列表框的样式
        BorderStyle = 0 //设置列表的边框
        GridLines = true //设置列表的表格线
        Checkboxes = checkIt //设置列表的复选框
        FullRowSelect = true //选择某单元格时，是否选中整行
        for(var i = 0; i < iEnd; i ++)
        {
            //逐列添加表头
            ColumnHeaders.Add(i + 1, 'Col' + i, theTable.rows[0].cells[i]. innerText,
colWidth)
        }
        for(var i = 1; i < theTable.rows.length; i ++)
        {
            //逐列添加列表项
            myList.ListItems.Add( i, 'Key' + theTable.rows[i].cells[iId -1].
innerText.replace(" ", ""), theTable.rows[i].cells[0].innerText.replace(" ", ""))
            for(var j = 1; j < iEnd; j ++)
            {
                ListItems(i).SubItems(j) = theTable.rows[i].cells[j].innerText. replace(" ", "")
            }
        }
    }
}
</script>
<script language = 'javascript' for = 'myList' event = 'ItemClick(Item)'>
    var theValue = "" //定义选项变量
    theValue = Item.Text + Item.Key //设置选项内容
    for(i = 1; i <= Item.ListSubItems.Count; i ++)
    theValue = theValue + "\n" + Item.ListSubItems(i).Text //当前列表项内容
    myValue.value = theValue //在文本框中显示列表内容
</script>
<script language = 'javascript'>
    document.write( "<object classid='clsid:BDD1F04B-855B-11D1-B16A-00C0F0283628' style =
'width:" + (document.body.clientWidth - 150) + ";height:" + document.body.clientHeight + "'
id='myList'></object> ")
</script>
```

【运行效果】

Listview 浏览效果如图 6-14 所示。



图 6-14 ListView 浏览效果

【难点剖析】

本例的重点是 Ojbect 组件的使用。ListView 是 Microsoft 为 IE 浏览器提供的一个复杂列表框控件，可实现列表项的编辑、排序和显示。要引用此控件，必须在 Object 元素中指明该控件的“classid”属性，具体引用方法可参考代码。

第 7 章

键盘操作和状态栏特效

本章导读

状态栏通常用来显示页面的一些操作提示，如当鼠标指向链接时，状态栏会显示该链接的 URL。键盘可以使用功能键，替代按钮和鼠标的一些操作。本章介绍 Web 开发中常见的状态栏和键盘特效。

7.1 按功能键返回首页

【实例描述】

为了方便用户操作，可以为用户设置功能键，代替使用频率比较高的操作。本例介绍如何按“F8”键返回首页，默认首页是“http://www.google.com”。

【实现代码】

```
<script language="javascript">
function goHome()
{
    if(event.keyCode==119)                //判断按键
    {
        document.location.href="http://www.google.com"; //指定首页
    }
}
</script>
```

需要在 body 中添加判断按键的事件，代码如下所示：

```
<body onkeydown="goHome()">
```

【运行效果】

当按“F8”键时页面的切换效果如图 7-1 所示。

【难点剖析】

在使用功能键时，需要知道每个功能键的键值，F1 ~ F12 的键值分别是“112” ~ “123”。还要注意实现页面导航使用的是“location.href”属性。



图 7-1 按 F8 键的页面切换效果

7.2 Enter 键实现 Tab 键功能

【实例描述】

在设计 C/S 桌面程序时，用户按 Enter 键，光标会自动跳到下一个表格处。这样可以提高用户的工作效率。本例将介绍如何在 B/S 程序中实现此功能。

【实现代码】

```
<script language="javascript">
function changeFocus()
{
    if(event.keyCode==13)
        event.keyCode=9;
}
</script>
```

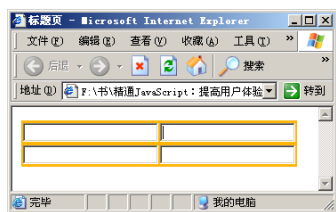


图 7-2 按 Enter 键后的界面

需要在 body 中添加一个包含文本框的 table, 其中的文本框需要调用上面的方法。

【运行效果】

按 Enter 键后的界面如图 7-2 所示。

【难点剖析】

本例的重点是要知道 Enter 和 Tab 的键值, 以及如何获取用户输入的键。用户输入的键通过 “event.keyCode” 获取, Enter 的键值是 “13”, Tab 的键值为 “9”。

7.3 Ctrl+Enter 提交数据

【实例描述】

在使用 QQ 提交数据的时候, 用户可以选择使用按 Enter 键提交, 或者使用按 “Ctrl+Enter” 组合键提交, 这样就方便了用户操作。本例学习使用 “Ctrl+Enter” 组合键提交数据。

【实现代码】

```
<script language=javascript>
//判断浏览器类型
ie = (document.all)? true:false
if (ie)
{
function ctlent(eventobject)
{
//获取用户输入的键值
if(event.ctrlKey && window.event.keyCode==13)
{this.document.form1.submit();}
}
}
</script>
```

需要在 body 中添加一个名为 form1 的窗体, 代码如下所示:

```
<form action="http://www.baidu.com" method="get" name="form1">
<textarea cols="50" name="Content" rows="10" wrap="virtual" onkeydown= "ctlent()">
Ctrl+Enter 提交内容
</textarea>
<input type=Submit value="Submit" name=Submit>
</form>
```

【运行效果】

实例的运行效果如图 7-3 所示。

【难点剖析】

本例的重点是如何判断用户同时选择了 Ctrl 和 Enter 键。在 JavaScript 中, “event.ctrlKey”

用来判断用户是否选择了 Ctrl 键，“event.keyCode==13”用来判断用户是否选择了 Enter 键。通过判断是否同时满足这两个条件，来判断用户的选择。

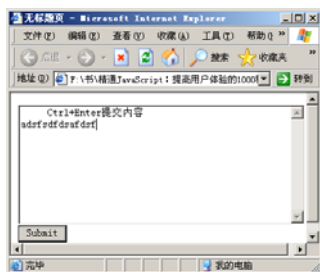


图 7-3 实例的运行效果

7.4 IE 中屏蔽退格键（BackSpace）

【实例描述】

用户在 textarea 中输入内容时，不允许使用退格键。本例学习如何实现这种屏蔽。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="javascript">
document.onkeydown = function()                                //用户的按键事件
{
    if(event.keyCode == 8)                                     //如果按下的是退格键
    {
        if(event.srcElement.tagName.toLowerCase() == "textarea") //如果是在 textarea 内
            event.returnValue = false;                          //不执行任何操作
    }
}
</script>
<input type="text">在这里看看能不能使用退格键<p>
<textarea cols=80 rows=10>在这里看看能不能使用退格键</textarea>
</body>
</html>
```

【难点剖析】

本例的重点是使用“event.keyCode”来判断用户的按键。“onkeydown”是键盘操作触发的窗体事件，退格键的键值为“8”。表达式“event.returnValue=false”表示当用户按退格键时，不执行任何操作。

7.5 屏蔽键盘所有键

【实例描述】

很多网页为了防止用户非法输入，将键盘屏蔽掉，只允许使用屏幕上的软键盘。本例学习如何屏蔽键盘的所有键。



【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
//重写文档的按键事件
function document.onkeydown(){
    event.keyCode = 0;                //忽略所有键
    event.returnValue = false;        //不执行按键操作
}
</script>
</head>
<body>
<input type="text" name="txt1" value="">
</body>
</html>
```

【难点剖析】

本例的重点是捕获网页中的按键操作。“onKeyDown”事件用来捕获用户的按键操作，“event.returnValue=false”表示不执行用户的按键操作。

7.6 JavaScript 捕获方向键

【实例描述】

本例可以应用在游戏或财务系统中，判断用户使用了哪个方向键。

【实现代码】

```
<HTML>
<HEAD>
<title>反选</title>
<script language="javascript">
function check()
{
    if(event.keyCode == "37")
        alert("您选择了左键！");
    if(event.keyCode == "38")
        alert("您选择了上键！");
    if(event.keyCode == "39")
        alert("您选择了右键！");
    if(event.keyCode == "40")
        alert("您选择了下键！");
}
</script>
</HEAD>
<BODY onkeydown="check()">
</BODY>
```

</HTML>

【难点剖析】

本例的重点是“event.keyCode”的使用。event 对象用来捕获用户事件，“keyCode”属性表示用户按下的键的键值。方向键的键值各不相同，具备唯一性。

7.7 状态栏变化信息

【实例描述】

状态栏除可以实现文本滚动显示功能外，还可以通过数组实现信息的变化显示。本例学习如何在状态栏变化显示一组信息。

【实现代码】

```
<script language="JavaScript">
//设置显示速度等变量
var speed = 10                                //显示字与字的间隔
var pause = 1500                               //显示第二组的间隔
var timerID = null
var statusRun = false
//设置要提示的文本数组
var ar = new Array()
ar[0] = "这是第一组"
ar[1] = "这是第二组"
ar[2] = "这是第三组"

var message = 0
var state = ""
clearState()
//停止显示——也可作初始化显示
function stopStatus() {
    if (statusRun)
        clearTimeout(timerID)
    statusRun = false
}
//开始显示
function startStatus() {
    stopStatus()
    showStatus()
}
//初始化开始数据
function clearState() {
    state = ""
    for (var i = 0; i < ar[message].length; ++i) {
        state += "0"
    }
}
```



```

}
//在状态栏显示信息
function showStatus() {
    //判断是否显示下一组
    if (getString()) {
        message++
        if (ar.length <= message)
            message = 0
        clearState()
        timerID = setTimeout("showStatus()", pause)//间隔 1500 毫秒显示下一组
        statusRun = true
    } else {
        var str = ""
        for (var j = 0; j < state.length; ++j) {
            str += (state.charAt(j) == "1") ? ar[message].charAt(j) : " "
        }
        window.status = str
        timerID = setTimeout("showStatus()", speed) //间隔 10 毫秒显示下一字
        statusRun = true
    }
}
//用来判断显示组还是字的方法
function getString() {
    var full = true
    //此循环判断是否当前组数据没有显示完成
    for (var j = 0; j < state.length; ++j) {
        if (state.charAt(j) == 0) //初始时默认 state 都为 0
            full = false
    }
    //full 为真时，表示当前组数据显示完成
    if (full)
        return true
    while (1) {
        var num = getRandom(ar[message].length)
        if (state.charAt(num) == "0")
            break
    }
    state = state.substring(0, num) + "1" + state.substring(num + 1, state.length)
    return false
}
//随机数的读取
function getRandom(max) {
    return Math.round((max - 1) * Math.random())
}
</script>

```

【运行效果】

状态栏的显示效果如图 7-4 所示。

【难点剖析】

本例的重点在于如何判断第一组已经显示完毕，如何在第一组显示完毕后再调用第二组。本例通过“getString”方法判断当前组是否显示完毕，然后通过“setTimeout”定时器，继续调用后面的组。

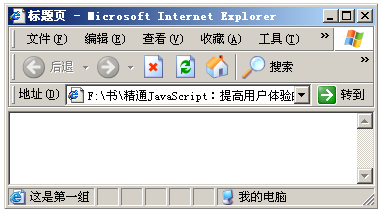


图 7-4 状态栏的显示效果

7.8 状态栏的跑马灯效果

【实例描述】

跑马灯效果一般指文本的逐字滚动或逐字显示。本例学习如何在状态栏显示文本，并实现文本的跑马灯效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <SCRIPT Language="JavaScript">
    var msg="欢迎光临北京红十字会，期待志愿者的参与!";
    var interval = 400;
    seq = 0;
    function marquee()
    {
      len = msg.length;
      window.status = msg.substring(0, seq+1);           //逐字增加文本
      seq++;
      if ( seq >= len ) { seq = 0 };                     //如果文本已经显示完，则从头开始
      window.setTimeout("marquee();", interval);
    }
  </SCRIPT>
</head>
<body onload="marquee()">
</body>
</html>
```

【运行效果】

状态栏的跑马灯效果如图 7-5 所示。

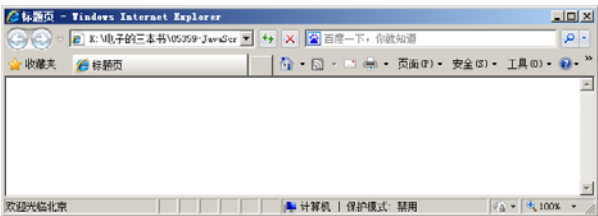


图 7-5 状态栏跑马灯效果



【难点剖析】

本例的重点有两个：在状态栏显示文字、以跑马灯效果显示文字。在状态栏显示文字用“window.status”，而跑马灯效果是借助“substring”方法，根据定时器的时间逐字显示文本。

7.9 状态栏缩放文字

【实例描述】

缩放文字指的是将一组文字慢慢地聚拢在一起。本例演示如何在状态栏实现这种效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
myTime = 50;
valtmp = 10;
msg = "欢迎大家来北京旅游! ";           //要显示的文本
amsg = new Array(33);
amsg[0] = msg;
blnk = " ";                               //设置一段空格
for (i = 1; i < 32; i++) {
    b = blnk.substring(0, i);
    amsg[i] = " ";
    for (j = 0; j < msg.length; j++)      //循环用空格间隔字符串
        amsg[i] = amsg[i] + msg.charAt(j) + b; //形成最终的字符串数组
}
function viewMsg() {
    if (valtmp > -1) str = amsg[valtmp];
    else str = amsg[0];
    if (valtmp-- < -40) valtmp = 31;
    status = str;                          //依次展示字符串数组
    clearTimeout(myTime);
    myTime = setTimeout("viewMsg()", 150); //循环执行当前方法
}
</script>
<body OnLoad="viewMsg()">
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是字符串数组的创建。“msg”变量用来保存要显示的文字，“amsg”数组用来保存掺杂了一些空格的字符串文本。代码中通过一个“setTimeout”定时器，循环执行“viewMsg”方法实现文本的慢慢合并。

7.10 状态栏文字来回显示

【实例描述】

网页经常使用状态栏来提示信息，本例用两个简单的 JavaScript 方法，实现文字来回显示的效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var Message="欢迎购买 JavaScript 实例大全，希望您能继续支持！";
var place=1;
//文字显示的方法
function scrollIn()
{
//从第一个文字开始在状态栏显示
window.status=Message.substring(0, place);
if (place >= Message.length) {
    place=1;
    window.setTimeout("scrollOut()",300);           //间隔 300 毫秒，循环收回文字
} else {
    place++;
    window.setTimeout("scrollIn()",50);             //间隔 50 毫秒，循环显示文字
}
}
//文字收回的方法
function scrollOut()
{
window.status=Message.substring(place, Message.length);
if (place >= Message.length) {
    place=1;
    window.setTimeout("scrollIn()", 100);           //间隔 100 毫秒，循环显示文字
} else {
    place++;
    window.setTimeout("scrollOut()", 50);           //间隔 50 毫秒，循环收回文字
}
}
</SCRIPT>
</head>
<body bgcolor="#eaf4d9" onLoad="scrollIn()">
</body>
</html>
```



【运行效果】

状态栏文字来回显示的效果如图 7-6 所示。



图 7-6 状态栏文字来回显示的效果

【难点剖析】

本例的重点是“scrollIn”和“scrollOut”两个方法。“scrollIn”用来显示文本，当文本显示完毕后，调用“scrollOut”隐藏文本。程序依次循环调用“scrollIn”和“scrollOut”，实现文本来回显示的效果。

7.11 交替闪烁的状态栏

【实例描述】

状态栏可以显示一些提示信息。本例介绍如何以闪烁的特殊效果，在状态栏显示提示信息。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
var msg = "欢迎来到北京";
var speed = 1000;
var control = 1;
function msgFlash()
{
    if (control == 1)
    {
        window.status=msg;
        control=0;
    }
    else
    {
        window.status=" ";
        control=1;
    }
    setTimeout("msgFlash()",speed);
}
</script>
</head>
<body onload="msgFlash()">
</body>
</html>
```

【难点剖析】

本例的重点是闪烁效果的实现。闪烁通过定时器控制，每隔一段时间就调用“msgFlash”方法，在状态栏中显示信息。“control”变量用来控制是否显示提示信息，如果“control”为“1”

则显示，如果为“0”则不显示指定的信息。通过信息的显示与不显示实现文本的闪烁效果。

7.12 状态栏的分解显示文本特效

【实例描述】

在状态栏中显示一组文本，同时在显示时实现分解显示的效果。可运行代码了解分解显示的特效。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var str=""
var direction="left" //文字移动的方向
function craAry(n)
{
    for (var i = 0; i < n; i++) {this[i] = 0} //初始化数组
    return this
}
var js_mult1=3141
var js_mult2=5821
var js_m1=100000000
var js_m2=10000
var js_iseed=0
var js_iseed1=0
var js_iseed2=0

function ArrayRandom(n) //获取一个随机数
{
    if (js_iseed == 0)
    {
        now = new Date()
        js_iseed = now.getHours() + now.getMinutes() * 60
            + now.getSeconds() * 3600
    }
    js_iseed1 = js_iseed / js_m2
    js_iseed2 = js_iseed % js_m2
    var tmp = (((js_iseed2 * js_mult1 + js_iseed1 * js_mult2) % js_m2) *
        js_m2 + (js_iseed2 * js_mult2)) % js_m1
    js_iseed = (tmp + 1) % js_m1
    return (Math.floor((js_iseed/js_m1) * n))
}

msgArray = craAry(5) //初始化要显示的文本信息
msgArray[0] = "伦敦奥运会欢迎您的参与"
```



```

msgArray[1] = "上海世博会欢迎您的参与"
msgArray[2] = "为奥运会作贡献"
msgArray[3] = "为世博会作贡献"
msgArray[4] = "当一个合格的志愿者";
msg = ""
dmsg = ""

function viewMsg()
{
    if (msg == dmsg)
    {
        msg = msgArray[ArrayRandom(5)]           //获取数组中的一段随机文本
        dmsg = ""
        for (var ii = 0; ii < msg.length; ii++) dmsg += " "
        viewtime = window.setTimeout('viewMsg()',100)
        return true
    }
    var ii = ArrayRandom(msg.length)
    var astr = dmsg.substring(0, ii)              //间隔式地截取显示文本中的字符
    var bstr = msg.substring(ii, ii+1)
    var cstr = dmsg.substring(ii+1, dmsg.length)
    dmsg = astr+bstr+cstr
    window.status = dmsg                         //状态栏中显示文本信息
    viewtime = window.setTimeout('viewMsg()',50)  //循环显示状态栏中的信息
    return true
}
</SCRIPT>
</head>
<body bgcolor="#fef4d9" onload="viewtime = window.setTimeout('viewMsg()',500);">
</body>
</html>

```

【难点剖析】

本例中的重点是文本数组的随机获取，以及分解字符。随机获取文本数组使用“ArrayRandom”方法，其中使用“Math.floor”方法来获取一个整数。分解字符使用的是字符串对象的“substring”方法，此方法用来截取从指定位置开始到指定位置结束的字符串。

7.13 状态栏文字从右弹出

【实例描述】

状态栏的文字可以从不同方向弹出，实现不同的样式。本例实现从右边弹出文字的效果。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>

```

```

<title>标题页</title>
<script language="JavaScript">
    var MESSAGE="支持伦敦奥运会，支持上海世博会 "; //状态栏显示的文本
//弹出文本的主要方法
function snapIn(jumpSpaces,position)
{
    var msg = MESSAGE;
    var out = "";
    //循环读取文本信息
    for (var i=0; i<position; i++)
        {out += msg.charAt(i);}
    //生成100个空格
    for (i=1;i<jumpSpaces;i++)
        {out += " ";}
    //依次读取文本信息
    out += msg.charAt(position);
    //在状态栏显示文本信息
    window.status = out;
    //实现从右边弹出文本的方法
    if (jumpSpaces <= 1) {
        position++;
        if (msg.charAt(position) == ' ')
            {position++; }
        jumpSpaces = 100-position;
    }
    else if (jumpSpaces > 3)
        {jumpSpaces *= .75;}
    else
        {jumpSpaces--;}
    if (position != msg.length) {
        var cmd = "snapIn(" + jumpSpaces + "," + position + ")";
        scrollID = window.setTimeout(cmd,scroll.delay); //根据定时器，循环弹出文本
    } else {
        window.status="";
        jumpSpaces=0;
        position=0;
        cmd = "snapIn(" + jumpSpaces + "," + position + ")";
        scrollID = window.setTimeout(cmd,scroll.delay);
        return false; }
    return true;
}
snapIn(100,0);
</script>
</head>
<body>
</body>

```

</html>

【运行效果】

状态栏文字从右边弹出的效果如图 7-7 所示。

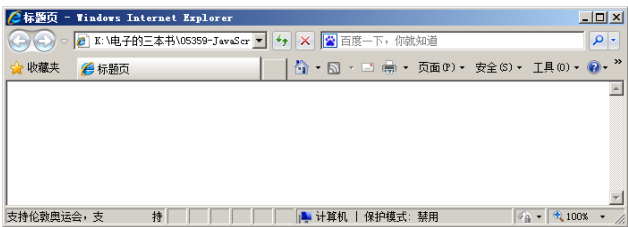


图 7-7 状态栏文字从右边弹出的效果

【难点剖析】

本例的重点是状态栏的文字显示。IE 中的状态栏使用 window 对象的“status”属性。从右边弹出文字主要是依靠位置的变化，使用“100-position”的方法从右边往左边显示文字。

7.14 状态栏中文字从中间分开显示

【实例描述】

状态栏的文字特效有很多，本例学习如何将文本由中间向两边展开并滚动显示。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var phrase = "伦敦奥运会，欢迎各路英雄和志愿者!! 期待您的参与! "; //要显示的信息
var lenPhrase = phrase.length; //信息的长度
var phraseOut = "";
var pause = 100; //间隔时间
var i=0;
var j=0;
function statusTxt()
{
    i++;
    phraseOut = "";
    for (j=1; j<=(lenPhrase/2)-i; j++) { //添加一半内容的空格
        phraseOut += " ";
    }
    for (j=1; j<=i; j++) {
        phraseOut += phrase.charAt(j-1); //开始从中间往左显示文本
    }
    for (j=i; j>=1; j--) {
        phraseOut += phrase.charAt(lenPhrase-j); //开始从中间往右显示文本
    }
}
```

```

    }

    window.status = phraseOut;           //在状态栏中输出
    if (i<lenPhrase/2) {                 //还没到一半的时候，循环执行
        setTimeout("statusTxt()",pause);
    }
}
</SCRIPT>
</head>
<body bgcolor="#fff4d9" OnLoad="statusTxt()">
</body>
</html>

```

【难点剖析】

本例的重点是获取中间位置。通过“lenPhrase/2”获取文本信息的中间位置，然后使用“charAt(j-1)”获取文本信息的一个字符，从中间位置逐个显示文本。“charAt”方法是 JavaScript 中字符串对象的常用操作，表示从指定位置获取一个字符。

7.15 屏蔽掉 IE 自带的功能键

【实例描述】

IE 自带了很多功能键，如按 F5 键可以刷新页面。但如果这些功能键与游戏网站的一些功能相冲突，该如何屏蔽 IE 自带的功能键呢？

【实现代码】

```

<html>
<head>
<title>Untitled</title>
<script>
document.onkeydown=noway;           //绑定窗体加载事件
function noway(){
    if(event.keyCode==116){         //通过键值判断是否是 F5
        event.keyCode=0;
        event.returnValue=false;   //不进行任何操作
    }
}
</script>
</head>
<body>
按 F5 测试是否能刷新页面
</body>
</html>

```

【难点剖析】

本例中通过“keyCode”来判断用户按下的是否是 F5 键。F5 键的键值是“116”，如果用户按下此键，则设置“event.returnValue”值为“false”，表示不执行任何操作。

第 8 章 链接特效

本章导读

链接一般用来导航当前页面到其他页面。在 HTML 用“a”标签标识，有时也被称为“锚”。本章将介绍链接的一些特殊 CSS 样式，并学习如何为链接添加功能提示。

8.1 关闭窗口的链接

【实例描述】

很多网页提供了关闭链接，用来方便用户操作。本例将通过一个链接，实现关闭窗口的效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<a href="javascript:window.close()"><br>
关闭窗口</a>
</body>
</html>
```

【运行效果】

此操作在关闭窗口时会有一个提示，如图 8-1 所示。

【难点剖析】

JavaScript 中的 window 对象用来操作窗口，可以实现窗口的打开和关闭。“window.open”用来打开窗口，“window.close”用来关闭窗口。

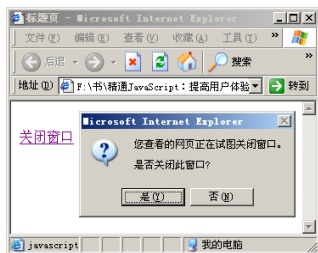


图 8-1 关闭窗口时的提示

8.2 不用 CSS 实现链接样式的变化

【实例描述】

通常样式的改变都需要使用 CSS 样式表完成。本例将使用 JavaScript 提供的样式属性，直接改变链接的样式。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<a href="http://www.baidu.com" style="color:navy"
onmouseover="this.style.fontWeight='bold';"
onmouseout="this.style.fontWeight='normal';">
把鼠标移动过来试验一下</a></body>
</html>
```

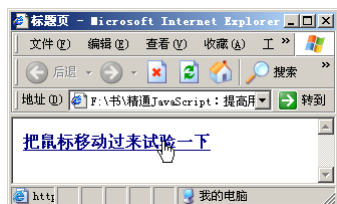


图 8-2 鼠标移动到链接上的效果

光标) 所指的对象, 然后使用 “style” 属性设置其样式。

【运行效果】

鼠标移动到链接上的效果如图 8-2 所示。

【难点剖析】

本例的重点是 JavaScript 中的 “style” 属性。此属性包含了对象 (如文本框、链接等) 常用的一些样式设置, 如字体、颜色、边框等。在 JavaScript 中, 可以直接使用 “this” 调用当前鼠标 (或

8.3 让链接没有下画线

【实例描述】

链接使用 “a” 标签实现, 默认链接包含下画线, 以区别普通的文本字符串。但有些情况下为了保持布局的完整, 不允许链接使用下画线。本例学习如何去掉这些默认的下画线。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style type="text/css">
    a {text-transform:none;text-decoration:none;}
    a:hover {text-decoration:underline}
</style>
</head>
<body>
<a href="http://www.google.com">我爱的搜索 </a>
</body>
</html>
```

【运行效果】

默认链接效果如图 8-3 所示。鼠标移动到链接上时的效果如图 8-4 所示。

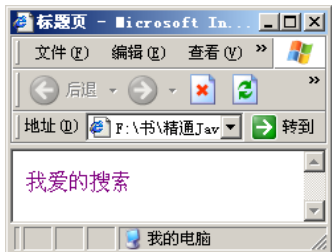


图 8-3 默认链接效果

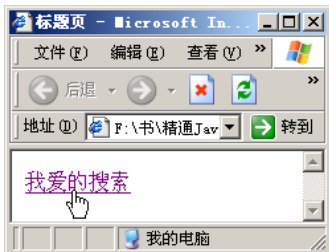


图 8-4 鼠标移动到链接上时的效果

【难点剖析】

本例的重点是 CSS 样式的使用。在样式表中, 使用 “a” 标签设置链接的属性, 其中 “text-decoration” 属性表示文本的下划线, “none” 表示禁止下划线。“a:hover” 表示鼠标移动到

链接上的效果,“underline”表示链接带下划线。

8.4 去掉超链接单击时的边框

【实例描述】

鼠标单击超链接时,链接周围会出现一个虚线边框。本例学习如何去掉这个边框。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style>
a {
    aa:expression(onfocus=function() {blur();});
}
</style>
</head>
<body>
单击下面的链接,测试是否会出现边框
<br />
<a href="#">单击这里</a>
</body>
</html>
```

【难点剖析】

本例的重点是为链接标签“a”设置样式,其中“onfocus”表示链接获得焦点时被触发的方法。“blur()”表示当链接被单击时,会自动失去焦点,这样就避免了边框的出现。

8.5 提取页面中所有链接

【实例描述】

提取链接有利于查找网站中的页面。本例学习如何提取页面中的所有链接。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript1.2">
function extractlinks()
{
    var links=document.all.tags("A")                //获取所有a元素
    var total=links.length                            //a元素的个数
    var win2=window.open( "", "", "menubar,scrollbars,toolbar" )
    //在新窗口中显示所有链接
```



```
win2.document.write("<font size='2'>一共有"+total+"个连接</font><br>")
for (i=0;i<total-1;i++){
win2.document.write("<font size='2'>"+links[i].outerHTML+"</font><br>")}
}
</script>
</head>
<body>
<p align="center">
<a href="expand.htm" OLDREF="#">首页</a>
<a href="expand.htm" OLDREF="#">链接 1</a>
<a href="expand.htm" OLDREF="#">链接 2 </a>
<a href="expand.htm" OLDREF="#">链接 3</a>
<a href="expand.htm" OLDREF="#">链接 4</a>
<a href="expand.htm" OLDREF="#">链接 5</a>
<a href="expand.htm" OLDREF="#">链接 6</a> </p>
<input type="button" onClick="extractlinks()" value="显示所有的连接">
</body>
</html>
```

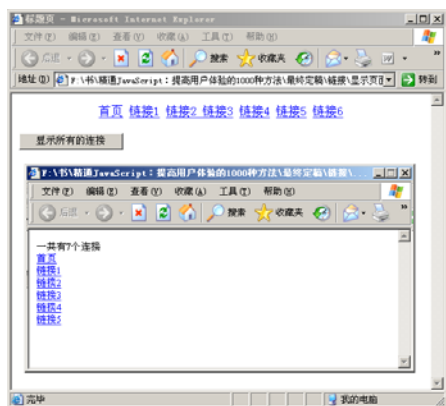


图 8-5 链接提取后的效果

【运行效果】

链接提取后的效果如图 8-5 所示。

【难点剖析】

获取页面中的所有链接，使用“document.all.tags(“A”)”，“tag”表示页面中的标记，链接的标记是“A”。判断链接的个数，使用“links.length”。

8.6 一个链接打开两个地址

【实例描述】

本例通过一个“<a>”链接，打开了两个 URL 地址。一个在新窗口中打开，一个在当前窗口打开。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function hrefClick(newWin, locationWin) {
window.open(newWin); //打开新的窗口
window.location = locationWin; //当前位置窗口也改变
}
</script>
</head>
<body>
```

```
<a href="javascript:hrefClick('http://www.google.net', 'http://www.baidu.net');">
打开页面</a></body>
</html>
```

【难点剖析】

本例的重点其实是两个参数的传递和“window.open”方法。“hrefClick”方法获取两个参数，“window.open”方法可以打开一个新的窗口，其中的参数就是所要打开的窗口的 URL。“window.location”属性是设置当前窗口的 URL。

8.7 为链接提供下拉菜单

【实例描述】

导航菜单是网站中常用的控件，为了简化操作，可以为链接设计一个菜单。本例学习如何实现链接的下拉菜单。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>无标题页</title>
    <script language="javascript">
        function listmenu()
        {
            document.all.sonMain.style.display=(document.all.sonMain.style. display==
'none')?'':'none';
        }
    </script>
</head>
<body>
    <div id="main" style="color:blue; margin:4px 4px 4px 60px;" onclick="listmenu()">
        <a href="#"> 我的网站<a></div>
        <div id="sonMain" style="display:none; margin:4px 4px 4px 70px;">
            <a href="#">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&关于开发语言</a><br>
            <a href="#">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&关于程序员前途</a><br>
            <a href="#">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&Microsoft 前进之路</a><br>
            <a href="#">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&Sun 前进之路</a><br>
            <a href="#">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&数据发展历史回眸</a><br>
            <a href="#">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&数据库的由来</a><br>
            <a href="#">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&Web2.0 时代的到来</a><br>
        </div>
    </body>
</html>
```

【运行效果】

下拉菜单的显示效果如图 8-6 所示。

【难点剖析】

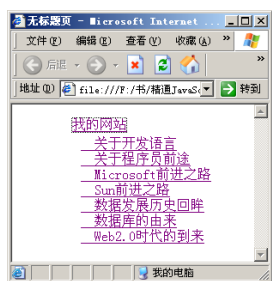


图 8-6 下拉菜单的显示效果
不同的只是表现形式。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type="button" value="最爱的搜索引擎" onClick="window.open('http://www.google.com',
'Sample', 'toolbar=no,location=no,directories=no,status=no,menubar=no,scrollbars=no,
resizable=yes,copyhistory=yes,width=790,height=520,left=0,top=0')" name="Input">
</body>
</html>
```

【难点剖析】

本例的重点是如何打开链接。代码中使用了“window.open”方法，此方法可以打开不同显示模式的窗口，如不带工具栏的、不允许最大化的等。“open”方法的第三个参数就是用来设置窗口显示模式的，其中“toolbar”表示工具栏，“menubar”表示菜单栏，“scrollbar”表示滚动条。

8.9 弹出鼠标所指的链接地址

【实例描述】

当用户查看网页中的链接时，有时需要了解这些链接的详细 URL 地址。为了方便用户操作，可将这些地址显示在状态栏中，也可以通过对话框的形式提示用户。本例将介绍对话框式的链接地址提示功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body onmouseover="if (event.srcElement.tagName=='A')alert(event.srcElement.href)">
<a href="http://www.google.com" >Google 搜索</a><br />
```

```
<a href="http://www.baidu.com" >百度搜索</a>
</body>
</html>
```

【运行效果】

弹出搜索链接的效果如图 8-7 所示。

【难点剖析】

本例的难点是 IE 对象的触发器“event.srcElement”。此触发器获取触发当前事件的控件，本例通过“tagName”属性判断此控件是否为链接。如果是，则显示链接的 URL 地址，代码为“event.srcElement.href”。

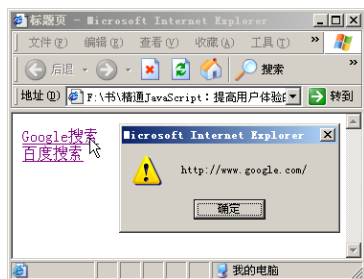


图 8-7 弹出搜索链接的效果

8.10 链接的注释

【实例描述】

当用户移动鼠标到某链接时，可以为链接提供简要说明。本例以滚动提示的方式，为链接设置注释。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script>
if (!document.layers&&!document.all)
event="test"
//显示提示的方法
function shownote(current,e,text)
{
//IE 浏览器的情况下
if (document.all&&document.readyState=="complete"){
//滚动显示提示内容
document.all.tip.innerHTML='<marquee style="border:1px solid black">'+text+ '</marquee>'
//根据鼠标位置设置提示位置
document.all.tip.style.pixelLeft=event.clientX+document.body.scrollLeft+10
document.all.tip.style.pixelTop=event.clientY+document.body.scrollTop+10
document.all.tip.style.visibility="visible"
}
//Netscape 浏览器的情况下
else if (document.layers){
document.tip.document.nstip.document.write('<b>'+text+'</b>')
document.tip.document.nstip.document.close()
document.tip.document.nstip.left=0
currentscroll=setInterval("scrolltip()",100)
```



```
document.tip.left=e.pageX+10
document.tip.top=e.pageY+10
document.tip.visibility="show"
}
}
//隐藏提示的方法
function hidenote()
{
if (document.all)
document.all.tip.style.visibility="hidden"           //隐藏 div 的显示
else if (document.layers){
clearInterval(currentscroll)
document.tip.visibility="hidden"
}
}
//提示信息的滚动方法
function scrolltip()
{
if (document.tip.document.nstip.left>=-document.tip.document.nstip. document.width)
document.tip.document.nstip.left-=5
else
document.tip.document.nstip.left=150
}
}
</script>
<div id="tip" style="position:absolute;clip:rect(0 150 50 0);width:150px; background-
color:#99FF99; top: 31px; left: 103px; visibility: hidden; height: 13px">
</div>
<a href="http://google.com" onMouseOver="shownote(this,event,'国外第一搜索引擎,欢迎使用') "
onMouseOut="hidenote()">国外搜索</a>
</body>
</html>
```

【运行效果】

链接提示的效果如图 8-8 所示。

【难点剖析】

本例的重点是 div 的隐藏和显示，还有提示信息的滚动功能。本例中提供三个方法：

“shownote”、“hidenote”和“scrolltip”。“shownote”方法显示 div 的提示信息，并通过“scrolltip”滚动显示提示文本。当鼠标离开链接时，使用“hidenote”方法隐藏注释。



图 8-8 链接提示的效果

【实例描述】

超链接使用锚“<a>”标签实现，通常不需要单击和双击事件，而是使用“href”属性，实现导航功能。本例的目的是学习如何为超链接绑定单击和双击事件。

8.11 为超链接同时绑定单击和双击事件

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script>
var dblNum=0;
function clk(cnum)
{
    dblNum=cnum;                //获取参数
    if(dblNum==1)window.setTimeout("if(dblNum!=2)window.alert('单击');dblNum= 0;",500);
                                //隔 500 毫秒后显示
    if(dblNum==2)alert("双击");    //显示双击
    return false; //不执行任何操作
}
</script>
<a onDbClick="return clk(2);" onClick="return clk(1);">双击</a>
</body>
</html>

```

【难点剖析】

本例没有特殊的功能，只是为一个控件同时添加单击和双击事件。注意在单击事件中，使用了一个定时器以延缓单击时的提示功能。

8.12 带链接的滚动字幕

【实例描述】

滚动文本是新闻显示的重要手段，如果要在滚动的文本中实现链接，则需要动态设置链接文本和链接地址。本例学习如何实现带链接的滚动字幕。

【实现代码】

```

<script language="JavaScript1.2">
var marqueeWidth=400 //设置 marquee 的宽度 (in pixels)
var marqueeHeight=20 //设置 marquee 的高度 (in pixels, 该参数只适用于 Netscape)
var speed=4          //设置 marquee 滚动的速度 (数值越大速度越快)
//设置 marquee 显示内容, 使用标准的 HTML 语法
var marqueeContents='<strong><big>欢迎支持中国搜索引擎 <a href="http://www. baidu.com">百
度一下</a> 找到自己想找的信息</big></strong></font>'
if (document.all)
    document.write('<marquee scrollAmount='+speed+'
style="width: '+marqueeWidth+' ">'+marqueeContents+'</marquee>')
function regenerate(){
    window.location.reload();                //重新加载页面
}

```



```
function regenerate2(){
    if (document.layers){
        setTimeout("window.onresize=regenerate",450);           // 窗体改变大小时重载
        initializemarquee();
    }
}
function initializemarquee(){
    //使用 nobr 控制显示的字符个数
    document.cmarquee01.document.cmarquee02.document.write('<no>'+marqueecontents+
'</no>');
    document.cmarquee01.document.cmarquee02.document.close();
    thelength=document.cmarquee01.document.cmarquee02.document.width; //获取层的宽度
    scrollit();                               //实现字体的滚动
}
function scrollit(){
    if (document.cmarquee01.document.cmarquee02.left>=thelength*(-1)){
        document.cmarquee01.document.cmarquee02.left-=speed;
        setTimeout("scrollit()",100);           // 定时器实现不停的调用
    }
    else{
        document.cmarquee01.document.cmarquee02.left=marqueewidth;
        scrollit();
    }
}
window.onload=regenerate2;
</script>
```

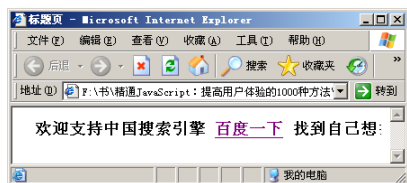


图 8-9 带链接的滚动字幕
用“scrollit”方法实现。

【运行效果】

带链接的滚动字幕如图 8-9 所示。

【难点剖析】

本例的重点主要包括如何动态添加链接和如何实现文本的滚动。代码中使用了一个全局变量“marqueecontents”来保存链接内容和地址。文本的滚动通过定时器不断地调

8.13 会跳舞的链接

【实例描述】

所谓跳舞，就是链接的颜色不断变化。本例使用定时器不断改变链接的颜色，实现链接的跳舞特效。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
```



```

<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function initArray() {
    for (var i = 0; i < initArray.arguments.length; i++) {
        this[i] = initArray.arguments[i];
    }
    this.length = initArray.arguments.length;    //获取初始化数组的参数个数
}
var colors = new initArray(                        //定义颜色数组
    "#ffffcc",
    "yellow",
    "green",
    "purple",
    "black",
    "tan",
    "gray");
delay = 0.6;                                     //延迟时间，单位为秒
link = 0;
vlink = 2;
function linkDance() {
    link = (link+1)%colors.length;                //设置链接的默认颜色
    vlink = (vlink+1)%colors.length;              //设置查看过的链接的默认颜色
    document.linkColor = colors[link];            //改变链接的颜色
    document.vlinkColor = colors[vlink];          //改变查看过的链接颜色
    setTimeout("linkDance()",delay*1000);         //设置定时器，实现颜色的定时变化
}
linkDance();
</script>
</head>
<body>
<a href="#" >这是一个跳舞的链接</a>
</body>
</html>

```

【难点剖析】

本例的重点是如何实现颜色的随机变化。代码中使用“initArray”，保存了一些颜色。使用“setTimeout”定时器不断执行“linkDance”方法，实现颜色的不断变化。

8.14 检测站点的链接速度

【实例描述】

判断一个站点的好坏，界面和速度都是关键因素。本例学习如何测试一个网站的链接速度。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>

```



```
<title>标题页</title>
</head>
<body>
<script>
indexArr=0 //连接网站的时间
setInterval("indexArr++",100) //设置循环的定时器
b=1
var autourl=new Array() //设置一个链接数组
autourl[1]="mail.163.com"
autourl[2]="www.263.net"
autourl[3]="www.sina.com.cn"
autourl[4]="www.baidu.com"
autourl[5]="www.cctv.com"

function writeBtn(){
    document.write("<form name=autof>") //输出一个提交窗体
    for(var i=1;i<autourl.length;i++)
        document.write("<input type=text name=txt"+i+" size=10 value=测试中.....> => <input
type=text name=url"+i+" size=40> => <input type=button value=GO onclick=window.open(this.
form.url"+i+".value)><br>")
    document.write("<input type=submit value=刷新></form>") //输出一个提交按钮
}
writeBtn()
function auto(url){
    document.forms[0]["url"+b].value=url
    if(indexArr>200)
        {document.forms[0]["txt"+b].value="链接超时"} //如果超时，则提示此信息
    else
        {document.forms[0]["txt"+b].value="时间"+indexArr/100+"毫秒"} //显示链接站点的时间
    b++
}
function run(){
    for(var i=1;i<autourl.length;i++) //循环测试链接站点的时间
        document.write("<img src=http://"+autourl[i]+"/"+Math.random()+ " width=1 height=1
onerror=auto('http://"+autourl[i]+"')>")
}
run()
</script>
</body>
</html>
```

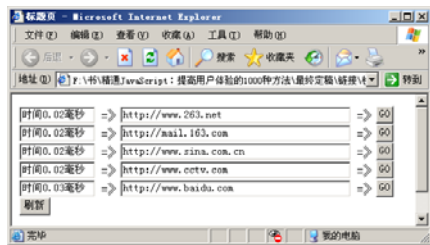


图 8-10 本例的运行效果

【运行效果】

本例的运行效果如图 8-10 所示。

【难点剖析】

本例的目的是测试网站的链接速度，其中使用了一个定时器，每隔 100 毫秒计时一次（“indexArr”变量自增），当站点连接完毕后，获取“indexArr”变量的值再除以 100，便是连接站点需要的毫秒值。

8.15 文本链接的渐变效果

【实例描述】

渐变效果属于页面的特效，可以通过一些链接或图像的渐变，增强用户的视觉体验。本例学习如何实现文本链接的渐变。

【实现代码】

```
<script language="javascript">
//内部变量
startColor = "#671700";           //定义链接颜色
endColor = "#D8D1C5";             //定义要渐变到最后的颜色
stepIn = 17;
stepOut = 23;
autoFade = true;                  //定义是否让所有的文本链接自动渐变
sloppyClass = false;              //特殊样式
hexa = new makearray(16);
for(var i = 0; i < 10; i++)
    hexa[i] = i;
hexa[10]="a"; hexa[11]="b"; hexa[12]="c";
hexa[13]="d"; hexa[14]="e"; hexa[15]="f";
//定义窗体事件
document.onmouseover = domouseover;
document.onmouseout = domouseout;
//初始设置
startColor = dehexize(startColor.toLowerCase());
endColor = dehexize(endColor.toLowerCase());
var fadeId = new Array();

function dehexize(Color)
{
    var colorArr = new makearray(3);
    for (i=1; i<7; i++){
        for (j=0; j<16; j++){
            if (Color.charAt(i) == hexa[j]){
                if (i%2 !=0)
                    colorArr[Math.floor((i-1)/2)]=eval(j)*16;
                else
                    colorArr[Math.floor((i-1)/2)]+=eval(j);
            }
        }
    }
    return colorArr;
}
//鼠标移动到链接时的颜色变换
function domouseover()
```



```
{
  if(document.all){
    var srcElement = event.srcElement;
    if ((srcElement.tagName == "A" && autoFade) || srcElement.className == "fade" ||
(sloppyClass && srcElement.className.indexOf("fade") != -1))
      fade(startColor,endColor,srcElement.uniqueID,stepIn);
  }
}
//鼠标移走时的颜色变换
function domouseout()
{
  if (document.all){
    var srcElement = event.srcElement;
    if ((srcElement.tagName == "A" && autoFade) || srcElement.className == "fade" ||
(sloppyClass && srcElement.className.indexOf("fade") != -1))
      fade(endColor,startColor,srcElement.uniqueID,stepOut);
  }
}
function makearray(n) //定义数组
{
  this.length = n;
  for(var i = 1; i <= n; i++)
    this[i] = 0;
  return this;
}
//考虑到颜色的16进制表示方法，实现转换
function hex(i)
{
  if (i < 0)
    return "00";
  else if (i > 255)
    return "ff";
  else
    return "" + hexa[Math.floor(i/16)] + hexa[i%16];
}
//定义颜色的方法
function setColor(r, g, b, element)
{
  var hr = hex(r); var hg = hex(g); var hb = hex(b);
  element.style.color = "#" + hr + hg + hb;
}
//实现颜色渐变的关键方法
function fade(s,e, element,step)
{
  var sr = s[0]; var sg = s[1]; var sb = s[2];
  var er = e[0]; var eg = e[1]; var eb = e[2];

  if (fadeId[0] != null && fade[0] != element){
```

```

setColor(sr,sg,sb,eval(fadeId[0]));
var i = 1;
while(i < fadeId.length){
    clearTimeout(fadeId[i]);
    i++;
}
for(var i = 0; i <= step; i++) {
    fadeId[i+1] = setTimeout("setColor(Math.floor(" +sr+ " * (( " +step+ " - " +i+ " ) / " +step+ " ) + " +er+ " * ( " +i+ "/" + step+ " )),Math.floor(" +sg+ " * (( " +step+ " - " +i+ " ) / " +step+ " ) + " +eg+ " * ( " +i+ "/" +step+ " )),Math.floor(" +sb+ " * (( " +step+ " - " +i+ " ) / " +step+ " ) + " +eb+ " * ( " +i+ "/" +step+ " )), "+element+");",i*step);
}
fadeId[0] = element;
}
</script>

```

需要在 body 中添加一个文本链接，代码如下所示：

```
<a href="http://www.google.com">最可爱的文本链接</a>
```

【运行效果】

实例的运行效果如图 8-11 所示，注意观察鼠标移动到链接上方时文本颜色的变化。

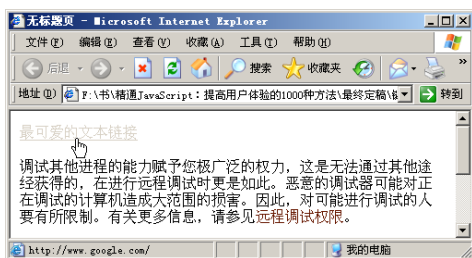


图 8-11 文本链接的渐变效果

【难点剖析】

本例的重点有 3 处：判断页面中的链接、定时变换颜色和颜色 RGB 的随机值。本例只对链接实现渐变，所以当鼠标移动时，需要判断指定的元素是否是链接，这通过“srcElement.tagName == "A"”判断。定时变换颜色是使用定时器结合代码中提供的“setColor”方法实现。颜色 RGB 值的获取非常关键，由于 RGB 是一个 16 进制的表示方法，所以代码中还使用了 Math 对象，并利用“Math.floor”来获取除法运算后的整数部分。

第 9 章 表格特效

本章导读

表格是 Web 1.0 时代常用的页面布局方式。虽然 Web 2.0 时代已经尽量改用 div 布局，但表格依然是很多网页开发者的首选。表格在 HTML 中用 table 作为标识，通常由行和列组成。本章将介绍如何动态设置表格的行列、如何遍历行列，以及如何动态编辑表格。

度和宽度这是测试高度和宽度内容

这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度内容内

容这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度内容

这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度内容内

容这是测试高度和宽度这是测试高度和宽度内容！

```
</div>
<td>
<td width="50%">
<div style="height:100px;overflow:auto;">
```

这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度这是测试高度和宽度内容内容内

容这是测试高度和宽度这是测试高度和宽度内！

```
</div>
<td>
</tr>
</table>
</body>
</html>
```

【运行效果】

本例的运行效果如图 9-2 所示。

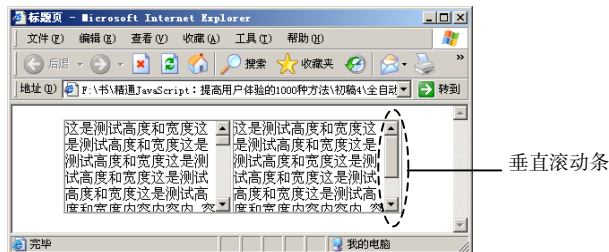


图 9-2 本例的运行效果

【难点剖析】

本例使用了 CSS 来控制单元格的高度和宽度。本例中将单元格的高度固定为 100px，然后使用了“overflow”来设置垂直滚动条。其值为“auto”，表示一旦内容超过单元格的高度，则自动出现滚动条。

9.3 突出的表格

【实例描述】

默认表格的样式是平的，本例提供一种样式，让表格看起来具有三维效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```



```
<head>
<title>标题页</title>
</head>
<body>
  <table border="1" cellpadding="2" cellspacing="4" bgcolor=#CCCC66 align=center>
    <tr>
      <td bgcolor=#CCCCFF bordercolorlight="#808080" bordercolordark= "#FFFFFF">第一行第
一列</td>
      <td bgcolor=#CCCCFF bordercolorlight="#808080" bordercolordark= "#FFFFFF">第一行第
二列</td>
    </tr>
    <tr>
      <td bgcolor=#CCCCFF bordercolorlight="#808080" bordercolordark= "#FFFFFF">第二行第
一列</td>
      <td bgcolor=#CCCCFF bordercolorlight="#808080" bordercolordark= "#FFFFFF">第二行第
二列</td>
    </tr>
  </table>
</body>
</html>
```

【运行效果】

表格突出的效果如图 9-3 所示。

【难点剖析】

本例的重点是样式的应用，其中列的“bordercolorlight”属性用来设置亮边框颜色，“bordercolordark”属性用来设置边框的背光颜色。

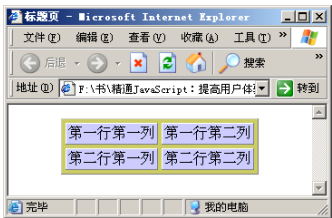


图 9-3 表格突出的效果

9.4 让表格有提示信息

【实例描述】

提示信息有很多种特效，如渐隐的、滚动的和幻灯片式的，但 HTML 提供一种标准的表格提示信息方法，是信息提示解决方案中最简单的。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<table border="1" width="100%">
<tr title="第一行"><td>表格 1</td><td>表格 2</td></tr>
<tr title="第二行"><td>表格 3</td><td>表格 4</td></tr>
<tr title="第三行"><td>表格 5</td><td>表格 6</td></tr>
```

```
</table>
</body>
</html>
```

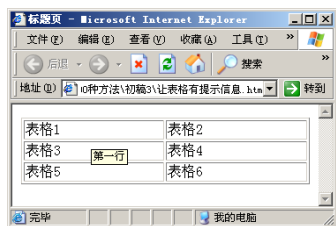


图 9-4 表格的提示效果

【运行效果】

表格的提示效果如图 9-4 所示。

【难点剖析】

本例的重点是表格中行的“title”属性，它用来显示一个标签式的提示效果。注意此属性和“caption”属性不同，“caption”属性用来设置表格的标题。

9.5 闪亮的表格边框

【实例描述】

可使用表格的样式属性“style”设置边框的特性。本例通过 JavaScript 设置表格的边框属性，实现边框的闪亮效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
  <table border="0" width="280" id="tbl" style="border:3px solid green">
    <tr>
      <td>
        这是一个闪亮的表格边框！
      </td>
    </tr>
  </table>
  <script language="JavaScript">
    function flashTable()
    {
      if (!document.all) return //判断浏览器的类型
      if (tbl.style.borderColor=="green") //判断表格的颜色是否为绿色
        tbl.style.borderColor="red" //将颜色更改为红色
      else
        tbl.style.borderColor="green" //将颜色更改为绿色
    }
    setInterval("flashTable()", 400) //每隔 400 毫秒就更新颜色
  </script> </body>
</html>
```

【运行效果】

表格边框的闪亮效果如图 9-5 所示。

【难点剖析】

本例的重点是表格的“style”属性，“green”表示将表格的颜色设置为绿色。本例使用一个定时器，每隔 400 毫秒调用一次“flashTable”方法，此方法通过“borderColor”属性不断修改边框的颜色，以实现闪亮效果。

9.6 表格的宽度固定后内容自动换行

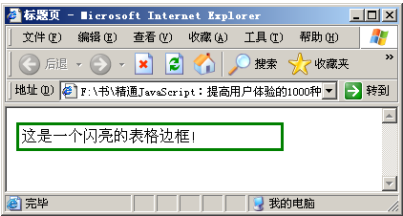
【实例描述】

很多情况下，为了不让表格被撑破，通常设置表格的宽度值为百分比。但有时为了控制页面的布局，也将表格设置为固定宽度。本例实现固定表格宽度时表格的列不被撑破。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<TABLE border="1" width="200">
<TR>
<TD>dfad</TD>
<TD style="word-wrap:break-word;width:100">
sfdaaaadsfaaaaafdasfasdfaaaaadfssadsfdasfsafsadfasdfsdfs</TD>
</TR>
</TABLE>
</body>
</html>
```

图 9-5 表格边框的闪亮效果



【运行效果】

本例的运行效果如图 9-6 所示。

【难点剖析】

本例的重点是表格样式的应用。“word-wrap”设置当前行的长度超过指定容器的边界长时是否换行。如果需要，词内换行“word-break”会按照英文单词实现换行。

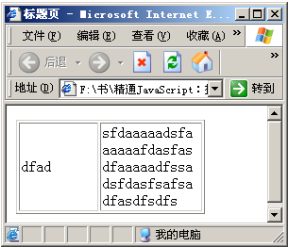


图 9-6 本例的运行效果

9.7 表格的排序

【实例描述】

为了可以清晰地显示表格的信息，用户可以对表格进行排序。下面以表格的升序为例，介绍如何实现排序功能。



【实现代码】

```
<html>
<head>
    <title>tree</title>
    <meta name="designer" content="csdn design team - meizz" />
</head>
<script language=javascript>
function show()
{
    var temp=new Array()
    var obj=document.getElementById("table1").childNodes[0] //获取当前页的表格
    var objs=obj.childNodes //获取表格的第一级子元素
                                //遍历表格子元素的

    for(var i=0;i<objs.length;i++)
    {
        //取表格的第三个元素
        temp[i]=new Array(objs[i].childNodes[2].innerText,objs[i])
    }
    temp.sort(function(a,b){return a[0]-b[0]}) //进行排序，自定义排序方法 sort
    for(var i=0;i<temp.length;i++)
        obj.appendChild(temp[i][1]) //将排序结果重新添加到表格中
}
</script>
<body>
<form name="myForm" method="post" action="/addcontent.jsp">
<table width="60%" border="0" cellspacing="0" cellpadding="0" align="center" id="table1">
    <tr>
        <td width="24%" align="center"><input type="checkbox" name="id" ></td>
        <td width="56%">用户体验的新技术</td>
        <td width="20%">3</td>
    </tr>
    <tr>
        <td width="24%" align="center"><input type="checkbox" name="id" ></td>
        <td width="56%">综合留言板</td>
        <td width="20%">2</td>
    </tr>
    <tr>
        <td width="24%" align="center"><input type="checkbox" name="id" ></td>
        <td width="56%">历史资料查询</td>
        <td width="20%">1</td>
    </tr>
</table>
<input name="button1" type="button" value="排序" onclick=show()>
</form>
</body>
</html>
```

【运行效果】

表格的初始运行效果如图 9-7 所示，排序后的效果如图 9-8 所示。

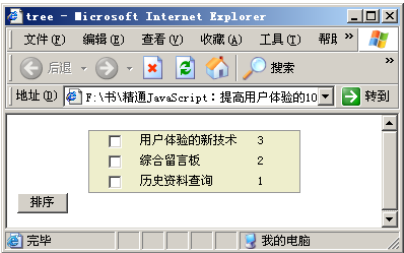


图 9-7 排序前

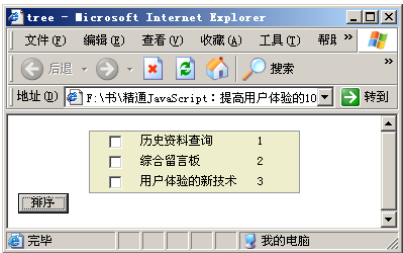


图 9-8 排序后

【难点剖析】

本例的重点是获取排序的值，以及如何进行排序。获取排序的值使用了 DOM 对象中的“childNodes”属性，如“childNodes[2].innerText”就是用来获取表格第二列的数据，然后使用“sort”方法实现排序。

9.8 表格的斜线

【实例描述】

如果要在网页中实现报表，则经常需要画一些斜线。本例学习一个非常简单的制作斜线的方法。

【实现代码】

```
<HTML>
<HEAD>
<TITLE> 新文档</TITLE>
</HEAD>
<script Language="javascript">
function aline(x,y,color)
{
    //实际画线的方法
    document.write("<img border='0' style='position: absolute; left: "+(x+20)+"; top: "+(y+20)+" ;background-color: "+color+" ' width=1 height=1>")
}
</script>
<body leftmargin=20 topmargin=20>
<TABLE border=0 bgcolor="000000" cellspacing="1" width=400>
<TR bgcolor="FFFFFF">
    <TD id="td1"> </TD>
    <TD>环境分</TD>
    <TD>人文分</TD>
    <TD>印象分</TD>
</TR>
```



```
<TR bgcolor="FFFFFF">
  <TD>北京</TD>
  <TD>80</TD>
  <TD>90</TD>
  <TD>80</TD>
</TR>
<TR bgcolor="FFFFFF">
  <TD>上海</TD>
  <TD>80</TD>
  <TD>70</TD>
  <TD>90</TD>
</TR>
<TR bgcolor="FFFFFF">
  <TD>天津</TD>
  <TD>80</TD>
  <TD>70</TD>
  <TD>70</TD>
</TR>
</TABLE>
<script>
function line(x1,y1,x2,y2,color)                                //画线的方法
{
  var tmp
  if(x1>=x2)
  {
    tmp=x1;
    x1=x2;
    x2=tmp;
    tmp=y1;
    y1=y2;
    y2=tmp;
  }
  for(var i=x1;i<=x2;i++)                                       //设置斜线的坐标
  {
    x = i;
    y = (y2 - y1) / (x2 - x1) * (x - x1) + y1;
    aline(x,y,color);
  }
}
line(td1.offsetLeft,td1.offsetTop,td1.offsetLeft+td1.offsetWidth,td1.offsetTop+td1.of
fsetHeight,'#000000') //指定画线位置
</script>
</BODY>
</HTML>
```

【运行效果】

表格的斜线效果如图 9-9 所示。

【难点剖析】

本例的重点是计算斜线的起始位置。“line”方法通过单元格“td1”的“offsetLeft”、“offsetTop”等属性，获取 4 个坐标点，然后整合成 X 坐标和 Y 坐标。最终通过“aline”方法，使用“img”控件完成了一条斜线的绘制。



图 9-9 表格的斜线效果

9.9 table 中的文字滚动

【实例描述】

文字可以在状态栏、页面中滚动，但和在表格中滚动的实现方法却不一样。本例学习如何实现 table 中的文字滚动。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<table border="1"><tr>
<td>第一列</td><td>
<marquee behavior="alternate" scrollamount="1" scrolldelay="100" width="230">第二列
</marquee>
</td><td>第三列</td>
</tr></table></body>
</html>
```

【运行效果】

table 中的文字滚动效果如图 9-10 所示。

【难点剖析】

本例的重点是 marquee 的“behavior”属性。此属性设置为“alternate”，表示 marquee 的内容到达容器的边缘后，按照相反的方向滚动回来，周而复始。“scrollamount”属性表示文本滚动的速度，值越大速度会越快。

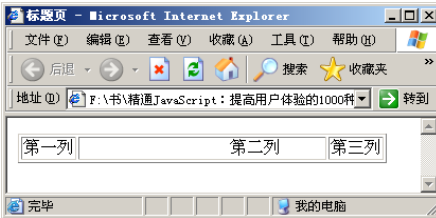


图 9-10 table 中的文字滚动效果

9.10 JavaScript 遍历 table 的行和列

【实例描述】

table 是网页中常用的布局方式。本例学习如何使用 JavaScript 处理 table 中的行和列。

【实现代码】

```
<HTML>
```



```
<head>
<SCRIPT LANGUAGE="JavaScript">
//遍历表格的每行、每列
function viewCell()
{
    var count=1; //在表格中显示的内容
    for (i=0; i < document.all.tbl.rows.length; i++) { //遍历每行
        for (j=0; j < document.all.tbl.rows(i).cells.length; j++) { //遍历行中的每一列
            document.all.tbl.rows(i).cells(j).innerText = count; //在单元格显示数值
            count++;
        }
    }
}
</SCRIPT>
</head>
<BODY onload="viewCell()">
<TABLE id=tbl border=1 width="200">
<TR><TH>&nbsp;</TH><TH>&nbsp;</TH><TH>&nbsp;</TH><TH>&nbsp;</TH></TR>
<TR><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD></TR>
<TR><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD></TR>
</TABLE>
</BODY>
</HTML>
```



【运行效果】

本例的运行效果如图 9-11 所示。

【难点剖析】

本例的重点是 table 中列和行的表示方法。“rows”和“cells”分别表示表格的行和列。如果要获取行数使用“rows.length”，获取列数使用“cells.length”。

图 9-11 本例的运行效果

9.11 表格按 Enter 键自动生成新行

【实例描述】

在 Excel 中，可以通过按 Enter 键将光标切换到下一行。本例将学习在 HTML 表格中，如何按 Enter 键自动添加新行。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function newRow()
{
```



```
if(event.keyCode=="13") //如果单击的是 Enter 键
{
    var row= tbl.insertRow(1); //添加行
    row.height="50"; //设置行高
    var cell1= row.insertCell(0); //添加列
    var cell2= row.insertCell(1); //添加列
    cell1.innerHTML="第二行" //指定列内容
    cell2.innerHTML="第二行第二列"
}
}
</script>
</head>
<body>
鼠标放到第二列，然后按回车键<br />
<table id="tbl" border="1"><tr><td>第一行第一列</td>
<td onkeypress="newRow()">第一行第二列</td></tr>
</table>
</body>
</html>
```

【运行效果】

本例的初始运行效果如图 9-12 所示。按 Enter 键后的效果如图 9-13 所示。

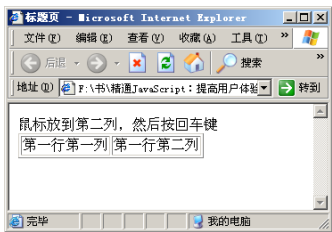


图 9-12 本例的初始运行效果



图 9-13 按 Enter 键后的效果

【难点剖析】

本例的重点有两个：动态创建行和列，以及捕获单元格的按键操作。创建行和列分别使用的是“insertRow”和“insertCell”方法。用“onkeypress”捕获单元格的按键操作，“event.keyCode”表示用户的按键值，其值为“13”时，表示按下的是 Enter 键。

9.12 单击单元格背景变色

【实例描述】

当单击单元格时，让单元格的背景发生变化，以突出用户的选择。本例学习如何动态改变单元格的背景色。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head>
<title>标题页</title>
<style>
.clickTD{
    background-color:#ff0000;
    text-decoration:none;}
</style>

<script language="javascript">
function colorChange(t)
{
    for (var i=0; i<t.rows.length; i++)                //遍历行
        for (var j=0; j<t.rows[i].cells.length; j++)    //遍历列
            t.rows[i].cells[j].className = t.rows[i].cells[j] == event.srcElement ?
'clickTD':''                                           //改变背景色
    }
</script>

</head>
<body>
<table border="1" onClick="colorChange(this)">
    <tr>
        <td>第一行</td><td>第一行第二列</td>
    </tr>
    <tr>
        <td>第二行</td><td>第二行第二列</td>
    </tr>
</table>
<p>
</body>
</html>
```

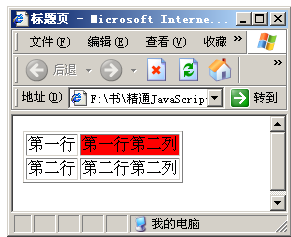


图 9-14 单击单元格后的效果

【运行效果】

单击单元格后的效果如图 9-14 所示。

【难点剖析】

本例的两个重点是如何遍历表格的行和列，以及如何使用三元运算符“?:”。用“rows.length”可以获取表格的行数，然后用“rows[i]”获取某行。用“cell.length”获取表格的列数，然后用“cells[j]”获取某列。三元运算符“?:”的使用语法如下所示：

条件==true?值 1:值 2

当满足条件时，将选择“?”后面第一个值，否则选择第二个。

9.13 单击表格某行后其他行隐藏

【实例描述】

本例是一个表格特效，为了突出用户选择的表格行，可以先隐藏其他的行。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT language="JavaScript">
var i
//viewTr 是要保留的行 ID,sumTr 是总的行数
function trdisplay(viewTr,sumTr){
    for (i = 0; i < sumTr; i++) {
        eval("document.getElementById('tr_' + (i+1) + ').style.display= 'none';");
        //隐藏所有的行
    }
    eval("document.getElementById('tr_' + viewTr + ').style.display='';");
    //显示指定要显示的行
}
</script>
</head>
<body>
<table width="70%" border="1" cellspacing="0" cellpadding="0">
    <tr id="tr_1" onClick="trdisplay(1,5)">
        <td>第一行</td>
    </tr>
    <tr id="tr_2" onClick="trdisplay(2,5)">
        <td>第二行</td>
    </tr>
    <tr id="tr_3" onClick="trdisplay(3,5)">
        <td>第三行</td>
    </tr>
    <tr id="tr_4" onClick="trdisplay(4,5)">
        <td>第四行</td>
    </tr>
    <tr id="tr_5" onClick="trdisplay(5,5)">
        <td>第五行</td>
    </tr>
</table>
</body>
</html>
```

【运行效果】

本例的初始效果如图 9-15 所示。隐藏其他行后的效果如图 9-16 所示。



图 9-15 本例的初始效果



图 9-16 隐藏其他行后的效果



【难点剖析】

本例要学习的是实现这种效果的技巧。用户单击某行后，首先是隐藏所有的行，然后通过传递的参数，判断用户选择的是哪行，最后再根据此行的 id 显示出该行。

9.14 单击表头实现表格排序

【实例描述】

为了让用户可以全方位地了解表格数据，有些网站提供单击表头实现表格排序的功能。本例将学习如何实现这种功能。

【实现代码】

```
<html>
<head>
<title>单击表格的表头，测试排序</title>
<script language="javascript">
function sortTable(sortType)
{
    var tb=document.getElementsByTagName("table")[0];           // 获取要排序的表格
    var arr=[];                                                    // 初始数组
    for (var i=1;i<tb.rows.length;i++)                            // 遍历表格中每一行
        arr.push(tb.rows[i].cells[sortType].innerText);          // 将列的数据添加到数组中
    sortType==0 ? arr.sort(function (a,b) {return a-b}) : arr.sort(); // 数组排序
    for (var j=1;j<tb.rows.length;j++)
        tb.rows[j].cells[sortType].innerText=arr[j-1];          // 输出排序后的结果
}
</script>
</head>
<body>
<table border="1">
    <tr><th onclick="sortTable(0);">按数字排序</th><th onclick="sortTable (1);">按字符串排
序</th></tr>
    <tr><td>563</td><td>张三</td></tr>
    <tr><td>425</td><td>abc</td></tr>
    <tr><td>452</td><td>历史</td></tr>
    <tr><td>548</td><td>北京</td></tr>
    <tr><td>42</td><td>xxx</td></tr>
    <tr><td>137</td><td>无 zhoi</td></tr>
</table>
</body>
</html>
```

【运行效果】

单击第一列表头后的效果如图 9-17 所示。

【难点剖析】

本例使用“th”设置表格的表头，并为其添加“onclick”事件。排序的技巧是先遍历表格的每行，然后将对应列的数据保存到数组中。如果是字符串的排序，则可以调用数组对象的“sort”方法，直接生成排序结果；如果是数值型数据，则使用自定义的“function (a,b){return a-b}”方法。

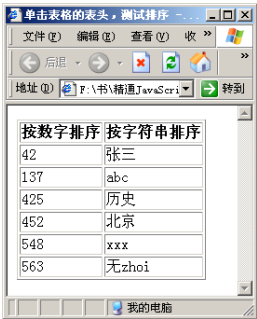


图 9-17 单击第一列表头后的效果

9.15 单击单元格显示行的详细信息

【实例描述】

有时由于页面布局的原因，有些表格只能显示其中的几列，为了让用户了解所有的行信息，可在单击某单元格的时候，通过 div 显示详细的行信息。本例学习如何实现这一功能。

【实现代码】

```
<html>
<head>
  <title></title>
</head>
<style type="text/css">
.div_class{
  margin: 0 auto;
  width: 200px;
  height: 17px;
  border: 1px solid #9EB1C0;
  padding: 1px;
}
</style>
<script type="text/javascript">
function displayDiv(obj){
  var myName = obj.innerText; //获取当前单元格内容
  var address = obj.parentNode.childNodes[1].innerText; //获取下一个单元格内容
  var phone = obj.parentNode.childNodes[2].innerText; //获取下下个单元格内容
  var objDiv = document.getElementById("div"); //获取要显示内容的 div
  objDiv.innerHTML = "姓名: " +myName+ "<br>联系地址: " +address+ "<br>电话: "+phone;
  objDiv.style.display="";
}

function hiddenDiv(){ //隐藏 div 的显示
  var obgDiv = document.getElementById("div");
  obgDiv.style.display="none";
}
</script>
<body>
<table>
```



```
<tr>
    <td>

<table width="400" border="1" cellpadding="1" cellspacing="1">
    <tr>
        <td>姓名</td>
        <td>联系地址</td>
        <td>电话</td>
    </tr>
    <tr>
        <td onclick="displayDiv(this)">张三</td>
        <td>北京</td>
        <td>010-88888888</td>
    </tr>
    <tr>
        <td onclick="displayDiv(this)">王武</td>
        <td>北京</td>
        <td>010-66666666</td>
    </tr>
</table>
<div id="div" class="div_class" style="display:none" ondblclick="hiddenDiv()">
</div>
    </td>
</tr>
</table>
</body>
</html>
```



图 9-18 单击单元格后的效果
定列数的表格。

【运行效果】

单击单元格后的效果如图 9-18 所示。

【难点剖析】

本例的重点是 DOM 对象的应用。在“displayDiv”方法中，使用“this”作为参数，获取当前单元格的内容，然后通过 DOM 对象的“parentNode”属性获取上级节点的对象，再使用“childNodes”依次获取上级节点的子节点。注意本例只针对固

9.16 表格设置为“100%”时获取表格的宽度

【实例描述】

为了控制表格的单元格不被撑破，通常使用百分比的形式设置表格的宽度，如“width=70%”。本例学习如何在表格设置为百分比宽度时，获取表格的实际宽度。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function getWidth()
{
    var tbl=event.srcElement;           //获取表格对象
    tbl.innerText=tbl.offsetWidth+'px';  //返回表格对象的实际宽度
}
</script>
</head>
<body>
<table width="100%" height="100" border="1" cellpadding="0" cellspacing="0" >
    <tr>
        <td id="tbl1" onMouseOver="getWidth()">光标移过来</td> <td id="tbl2" onMouseOver=
"getWidth()">光标移过来</td>
    </tr>
</table></body>
</html>
```

【运行效果】

本例的运行效果如图 9-19 所示。

【难点剖析】

本例的重点是“offsetWidth”属性，表示对象的可视宽度。实现原理是由“event.srcElement”属性获取当前操作的表格，然后使用“offsetWidth”属性获取当前单元格的宽度，最后使用“innerText”属性将宽度值显示在单元格内。

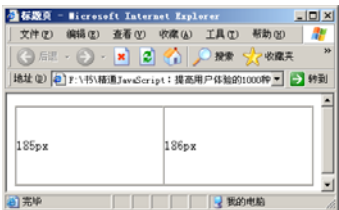


图 9-19 本例的运行效果

9.17 表格选中后变色

【实例描述】

在用户浏览表格时，为了突出显示表格内容，表格颜色会在鼠标移动到表格时发生变化。本例介绍如何使表格的颜色发生变化。

【实现代码】

```
<script language="javascript">
var searchResult=new Array();//鼠标滑过时显示背景色
//改变背景色和边框颜色的方法
function colorChange(table,color,color2)
{
    table.style.borderColor=color;
    table.style.backgroundColor=color2;
}
//鼠标移动过来后，更改颜色
function colorChange_on(e)
{
```



```

        if (document.all)
            source1=event.srcElement
        else if (document.getElementById)
            source1=e.target;
        if (source1.id=="mytable"){
            colorChange(source1,"#999999","#F8F8F6");
        }
        else{
            while(source1.tagName!="TABLE")
            {
                source1=document.getElementById? source1.parentNode : source1.
parentElement;

                if (source1.id=="mytable")
                    colorChange(source1,"#999999","#F8F8F6");
            }
        }
    }
    //鼠标移走后, 将颜色设置为白色
    function colorChange_off(e)
    {
        if (document.all)
            source2=event.srcElement
        else if (document.getElementById)
            source2=e.target
        if (source2.id=="mytable")
            colorChange(source2,"white","white")
        else{
            while(source2.tagName!="TABLE")
            {
                source2=document.getElementById? source2.parentNode : source2.
parentElement

                if (source2.id=="mytable")
                    colorChange(source2,"white","white")
            }
        }
    }
}
</script>

```

需要在 body 中添加表格, 并调用上面的方法, 代码如下所示:

```

<table width="80%" border="0" onMouseOver="colorChange_on(event)"
onMouseOut="colorChange_off(event)" id="mytable" >
    <TR>
        <TD>鼠标移动</TD>
        <TD>鼠标移动</TD>
    </TR>
    <TR>
        <TD>鼠标移动</TD>
        <TD>鼠标移动</TD>
    </TR>

```



```
<TR>
    <TD>鼠标移动</TD>
    <TD>鼠标移动</TD>
</TR>
<TR>
    <TD>鼠标移动</TD>
    <TD>鼠标移动</TD>
</TR>
</table>
```

【运行效果】

表格变色的效果如图 9-20 所示。

【难点剖析】

本例中如果要在 JavaScript 中获取表格，必须为表格指定“id”或“name”属性。获取鼠标所指元素是使用“event.srcElement”，获取元素后由其“id”判断它是否为指定的表格，如果是，则使用“style”属性修改元素的边框颜色和背景颜色。

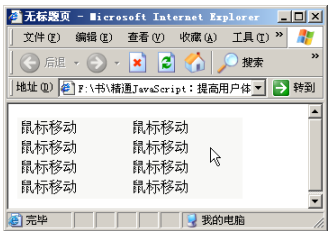


图 9-20 表格变色的效果

9.18 表格中隐藏下级表格

【实例描述】

div 层可以实现表单隐藏，但对于多个表单，使用表格隐藏手段反而更简单。本例学习如何使用表格隐藏。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<style type="text/css">
.color1 { color:#000080; cursor:hand;font-weight: bold; }
.color2 { color:#464646; display:none; }
</style>
<table onclick="divClick(this);">
<tr class="color1"><td>1、在工作流不同阶段，加载不同的表单？</td></tr>
<tr class="color2"><td>
    我现在已经把自定义表单实现了，加 WWF 以后，要做的东西就不多了<br>
    离可以作为工作流使用还有相当的开发工作量的</td></tr>
<tr class="color1"><td>2、为什么一定要先交预付款？</td></tr>
<tr class="color2"><td>
    请问这个是 text/x-scriptlet 这是个什么东东插件搜索服务显示是未知插件．．金箭．
<br>
    哪位有的下载啊我弄了半天也没弄好请高人帮帮忙啊~~谢谢~~<br>
```



```
款中扣除相应费用。同时，为保证您的用户购买域名、虚拟主机、FTP 空间、数据库、企业邮局能实时开  
2007 年 1 月 17 日 1:26 PM. 似乎是个 IE 下用的控件。</td></tr>  
<tr class="color1"><td>3、为什么公司的域名比较便宜？</td></tr>  
<tr class="color2"><td>  
    审核就把 delete from SMT_cp where SMT_id in(1,2,3)  
换成 update SMT_cp set 审核字段=1 where SMT_id in(1,2,3)就可以<br>  
SQL 语句可以直接 delete from SMT_cp where SMT_id in(1,2,3)删除</td></tr>  
</table>  
  
<SCRIPT language="javascript">  
function divClick(tb)  
{  
    var n = event.srcElement.parentNode.rowIndex; //获得父节点的行索引  
    if (n%2==0) //判断是否是被 2 整除的行-所有标题行都可以  
    {  
        with(tb.rows[n+1].style) display= display=="none" ? "block" : "none";  
        //让标题行的下一行显示  
    }  
}  
</SCRIPT>  
</body>  
</html>
```

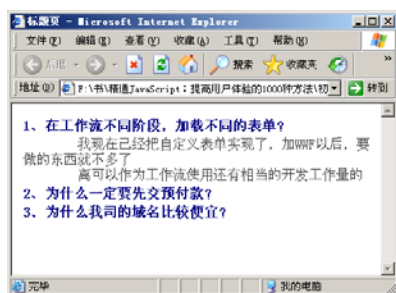


图 9-21 单击表格标题后显示的隐藏效果

【运行效果】

单击表格标题后显示的隐藏效果如图 9-21 所示。

【难点剖析】

本例中使用两种样式定义表格的标题行和内容行。当用户单击表格时，程序会通过“`n%2==0`”判断用户选择的行，如果能被 2 整除，说明用户单击的是标题行。当单击标题行时，使用表格的“`display`”属性控制内容行的显示。

9.19 表格自动下移

【实例描述】

当用户选中表格中某行时，可以改变此行的颜色。本例使用另外一种方法，实现行颜色的自动变化，并实现自上而下的行选择方式。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
</head>  
<body>  
<div id="tblView" style="border:#D3D3D3 1px solid;PADDING-LEFT: 1px;PADDING -top:
```

```

1px;width:339px;height:276px;overflow-x:hidden;overflow-y:scroll"></DIV>
</body>
<script language="javascript">
function bgChange(obj)
{
    for (var i=0; i<obj.parentElement.rows.length; i++)
        obj.parentElement.rows[i].bgColor = obj.parentElement.rows[i] == obj ? '#EFE9E9' :
'white'; //改变指定行的背景色
    }
    htm+="<table width=100% border=0 align=center cellpadding=0 cellspacing=0>"
    for(i=0;i<5;i++){ //动态输出 5 行表格
        o=i+1
        htm+="<tr id=td"+o+" height=10 style='CURSOR: hand;' onclick=bgChange(this);>"
        htm+="    <td style='font-size:12px;color:#6666f6;border-bottom:#D3D3D3 1px
solid;border-right:#D3D3D3 1px solid' width=17 height=10 align='center'>"+o+"</td>"
        htm+="    <td style='border-bottom:#D3D3D3 1px solid;border-right:#D3D3D3 1px solid'
width=218>&nbsp;  </td>"
        htm+="    <td style='border-bottom:#D3D3D3 1px solid' width=102>&nbsp;  </td>"
        htm+="</tr>"
    }
    htm+="</TABLE>"
    tblView.innerHTML=htm; //设置表格的内容
    b=0
    function setScroll(){ //调用表格滚动的方法
        if (b==o){b=0} //第一行
        b++;
        bgChange(document.getElementById("td"+b)); //调用改变背景色的方法
        setTimeout("setScroll()",2000); //每隔两秒调用一次
    }
    setScroll();
</script>
</html>

```

【运行效果】

本例的运行效果如图 9-22 所示。

【难点剖析】

本例的重点是动态添加表格，并设置表格中行的 id。然后通过一个循环，使用“bgColor”变量改变表格中行的背景色。注意行自动选择需要借助定时器“setTimeout”，本例中每隔两秒自动下移一行。

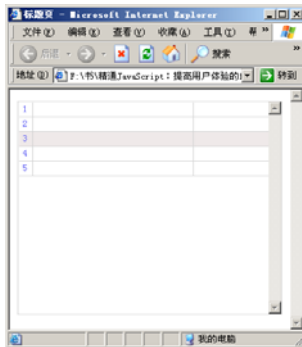


图 9-22 本例的运行效果

9.20 动态创建固定列数的表格

【实例描述】

本例学习如何动态创建列数固定的表格，并可以动态生成表格的内容。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<table width="100%" border="1" align="center" cellpadding="0" cellspacing="0">
<script language="javascript">
    for(var i=0;i<30;i++)                                //实现数值的遍历
    {
        if(i%3==0)                                        //是否能被 3 整除——固定列数的关键
            document.write(((i!=0)?"<\tr>":""+"<tr>")); //动态输出单元行
        document.write('<td align="center" bgcolor="#FFFFFF">第'+(i+1)+'个单元格<\td>')
                                                    //输出单元格及内容
    }
    document.write("<\tr>")
</script>
</table>
</body>
</html>
```



图 9-23 本例的运行效果

【运行效果】

本例的运行效果如图 9-23 所示。

【难点剖析】

本例的重点是如何固定列数。代码中使用“for”语句实现了一个 30 次的循环，然后使用“i%3==0”来判断是否换行。“i%==0”表示当前值是否能被 3 整除，如果能则换行，即凡遇到 3 的倍数时，就自动换行，这样就完成了固定三列的效果。

9.21 动态改变表格列宽

【实例描述】

每个控件都有很多的属性，如 id、name、样式、高度、宽度、value 等。项目中经常需要修改这些属性值，本例就学习如何获取页面元素的属性。

【实现代码】

```
<script language=javascript>
//鼠标按下时的方法
function MouseDownToResize(obj)
{
    obj.mouseDownX=event.clientX;                                //当前鼠标 x 坐标
    obj.pareneTdW=obj.parentElement.offsetWidth;                //父元素的宽度
```

```
obj.pareneTableW=table1.offsetWidth;           //表格的宽度
obj.setCapture();                               //捕获鼠标方法
}
//鼠标移动时的方法
function MouseMoveToResize(obj)
{
    if(!obj.mouseDownX) return false;           //判断是否已经按下
    var newWidth=obj.pareneTdW*1+event.clientX*1-obj.mouseDownX;
    if(newWidth>0)
    {
        obj.parentElement.style.width = newWidth;
        table1.style.width=obj.pareneTableW*1+event.clientX*1-obj.mouseDownX;
        //重新设计宽度
    }
}
//鼠标抬起时的方法
function MouseUpToResize(obj)
{
    obj.releaseCapture();                       //释放鼠标的捕获
    obj.mouseDownX=0;                          //鼠标抬起
}

</script>
```

【运行效果】

改变表格列宽时的效果如图 9-24 所示。

【难点剖析】

本例的重点是捕获鼠标的单击和移动事件，同时还要注意鼠标图形的变换。其中“SetCapture”主要用来捕获鼠标操作，而“ReleaseCapture”是用来释放鼠标的捕获的。



图 9-24 改变表格列宽时的效果

9.22 动态改变表格的行顺序

【实例描述】

Ajax 技术可以实现表格的多项特性，如拖动、编辑等，其实现的本质依然是 JavaScript 技术。本例将使用 JavaScript 实现表格的行拖动。

【实现代码】

```
<script language="javascript">
var beginMoving=false;           //判断是否移动的标识——移动开关
//鼠标按下时的操作
function MouseDownToMove(obj){
    obj.style.zIndex=1;          //样式
    obj.mouseDownY=event.clientY; //鼠标 y 坐标
```



```
obj.mouseDownX=event.clientX;           // 鼠标 x 坐标
beginMoving=true;                       // 开始移动
obj.setCapture();                       // 捕获鼠标操作
}
// 鼠标按下并移动时的操作
function MouseMoveToMove(obj){
    if(!beginMoving) return false;
    // 改变目标行的(x,y)坐标
    obj.style.top = (event.clientY-obj.mouseDownY);
    obj.style.left = (event.clientX-obj.mouseDownX);
}
// 鼠标抬起时的操作
function MouseUpToMove(obj){
    if(!beginMoving) return false;
    obj.releaseCapture();                // 释放对鼠标的捕获
    obj.style.top=0;
    obj.style.left=0;
    obj.style.zIndex=0;
    beginMoving=false;                  // 关闭移动开关
    var tempTop=event.clientY-obj.mouseDownY;
    var tempRowIndex=(tempTop-tempTop%20)/20; // 根据行高度获取行位置索引
    if(tempRowIndex+obj.rowIndex <0 )
        tempRowIndex=-1;
    else tempRowIndex=tempRowIndex+obj.rowIndex; // 实际的行索引
    if(tempRowIndex >= obj.parentElement.rows.length-1) tempRowIndex = obj.parentElement.
rows.length-1;
    obj.parentElement.moveRow(obj.rowIndex,tempRowIndex); // 移动行到指定位置
}
</script>
```

【运行效果】

本例初始运行效果如图 9-25 所示。行拖动后的效果如图 9-26 所示。

【难点剖析】

本例的重点是鼠标的三个方法：按下、移动和抬起，使用“setCapture”捕获鼠标的操作，一直到“releaseCapture”释放鼠标为止。最后根据行的高度判断行的位置索引，使用“move”方法移动行到指定位置。



图 9-25 初始运行效果



图 9-26 行拖动后的效果

9.23 动态生成包含合并单元格的表格

【实例描述】

表格的单元格可以使用“colSpan”或“rowSpan”属性来合并，但在动态创建表格时，使用这两个属性却很困难。本例将演示如何动态生成具有合并单元格的表格。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style>
TABLE{
    font-family: 宋体;
    font-size: 9pt;
    border-color:#7195c6;
    border-collapse :collapse;
    border-width:1px
}

td{
    border:#336699 1px solid;
    font-size:15px;
    color:#02027a
}
</style>

</head>
<body>
<script language="JavaScript">
var myData=[[ "姓名", "年龄", "年级", "地址,电话", "手机", "备注"],
[null,null,null, "历史", "地理", "88888888"],
[null,null,null,null, "本例", "66666666"],
[null,null, "一年级", "北京三环", "_", "33333333"],
[null,null, "二年级", "上海浦东", "_", "666666"],
[null, "12", "三年级", "广州深圳", "_", "33333222"],
[null,null, "四年级", "香港九龙", "_", "32432432"]]; //将所有数据绑定在数组中
var myDataT,rowcell=[]; //默认的两个空数组

document.body.appendChild((myDataT = document.createElement("TABLE"))); //在当前窗体中动态添加表格
for(var i=0; i<myData.length; i++){ //遍历表格中每项
    var atr = myDataT.insertRow(); //动态添加行
    for(var j=0; j<myData[i].length; j++){ //遍历某项中的所有数据
        if (myData[i][j]==null){ //如果值为空，则表示需要合并
```

```
        rowcell[j].rowSpan++; //使用 rowspan
    }
    else if(myData[i][j]=="_"){ //如果是“_”，则使用 colspan
        rowcell[j-1].colSpan++;
    }
    else{
        rowcell[j] = atr.insertCell(); //否则正常添加行
        rowcell[j].innerText=myData[i][j]; //显示行内容
    }
}
}
</script>
</body>
</html>
```



图 9-27 本例的运行效果

【运行效果】

本例的运行效果如图 9-27 所示。

【难点剖析】

本例使用 DOM 对象的一些方法实现表格的动态创建。“createElement”方法用来创建一个 HTML 标签，“appendChild”方法将创建的元素添加到文档中。“rowSpan”和“colSpan”用来合并表格。本例的数组“myData”对于合并行和列都有固定的数据，为“null”时合并行，为“_”时合并列。

9.24 用键盘上下键实现表格行的上下选择

【实例描述】

表格中行的选择可以通过单击鼠标完成，但追求速度的用户更愿意操作键盘。本例学习如何设置通过键盘的方向键，实现表格中行的上下选择。

【实现代码】

```
<html>
<head>
<title>键盘方向键控制表格</title>
</head>
<body onKeyDown="keyCheck();" >
    <table width="80" bgcolor="#FFFFFF" height="60" border="1" bordercolor=" #FFFFFF"
cellpadding="0" cellspacing="0">
        <tr>
            <td id="td1" width="80">第一行</td>
        </tr>
        <tr>
            <td id="td2" width="80">第二行</td>
        </tr>
```



```

<tr>
  <td id="td3" width="80">第三行</td>
</tr>
<tr>
  <td id="td4" width="80">第四行</td>
</tr>
<tr>
  <td id="td5" width="80">第五行</td>
</tr>
<tr>
  <td id="td6" width="80">第六行</td>
</tr>
</table>
<script language="javascript">
var tdIndex = 1; //获取当前行的索引变量
document.all.td1.style.backgroundColor='#3366aa'; //设置列 1 的背景色
function keyCheck() {
  if (window.event.keyCode==38) { //向上键
    for (var i=1;i<=6;i++) {
      eval("document.all.td"+i+".style.backgroundColor='#FFFFFF'"); //更改所有的行背景色
    }
    if (tdIndex<=1) {
      document.all.td1.style.backgroundColor='#3366aa'; //到顶端时，只第一行颜色改变
      alert('已到顶端');
      return false;
    }
    else {
      tdIndex -= 1; //行索引减小
      eval("document.all.td"+tdIndex+".style.backgroundColor='#3366aa'"); //改变行的背景色
    }
  }
  if (window.event.keyCode==40) { //向下键
    for (var i=1;i<=6;i++) {
      eval("document.all.td"+i+".style.backgroundColor='#FFFFFF'"); //更改所有的行背景色
    }
    if (tdIndex>=6) {
      document.all.td6.style.backgroundColor='#3366aa'; //到顶端时，只第一行颜色改变
      alert('已到底端');
      return false;
    }
    else {
      tdIndex += 1; //行索引增加
      eval("document.all.td"+tdIndex+".style.backgroundColor='#3366aa'");

```

//改变行的背景色

```
}  
}  
}  
</script>  
</body>  
</html>
```

【运行效果】

用键盘控制的效果如图 9-28 所示。

【难点剖析】

本例的难点就是判断用户按下了哪个方向键。使用“onKeyDown”捕获键盘的按键操作，然后通过“event.keyCode”判断按下的是否是上、下键，如果是则改变当前行的索引，并设置行的背景色。

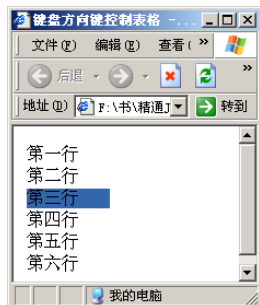


图 9-28 用键盘控制的效果

9.25 用 JavaScript 隐藏或显示表格列

【实例描述】

本例学习如何动态隐藏表格中的某一列，但并不是先删除再添加，而是使用列的“display”属性。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
</head>  
<body>  
<table border=1 cellpadding=0 cellspacing=0 width="80%">  
  <tr>  
    <td>第一列  
    </td>  
    <td>第二列  
    </td>  
    <td>第三列  
    </td>  
    <td>第四列  
    </td>  
  </tr>  
  <tr>  
    <td>第一列  
    </td>  
    <td>第二列  
    </td>  
    <td>第三列  
    </td>
```

```
<td>第四列
</td>
</tr>
</table>
<input type=button value==隐藏第三列 onclick=hide()>
<input type=button value==显示第三列 onclick=show()>
<script language=javascript>
    function hide()
    {
        var tr=document.getElementsByTagName('tr');           //获取所有的行
        var i;
        for(i=0;i<tr.length;i++)                               //遍历每一列
        {
            tr[i].getElementsByTagName('td')[2].style.display='none'; //隐藏第三列
        }
    }
    function show()
    {
        var tr=document.getElementsByTagName('tr');           //获取所有的行
        var i;
        for(i=0;i<tr.length;i++)                               //遍历每一列
        {
            tr[i].getElementsByTagName('td')[2].style.display='block'; //隐藏第三列
        }
    }
</script>
</body>
</html>
```

【运行效果】

本例的初始运行效果如图 9-29 所示。隐藏第三列后的效果如图 9-30 所示。

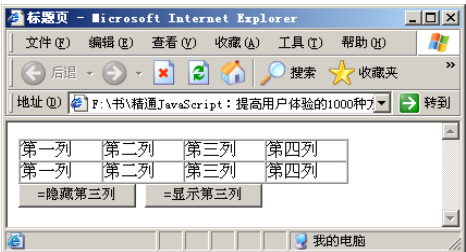


图 9-29 本例的初始运行效果



图 9-30 隐藏第三列后的效果

【难点剖析】

本例的重点有两个：如何获取指定的表格列，以及如何设置列隐藏。获取指定的列是通过遍历行的方法，使用“for”语句遍历表格中所有的行。设置列隐藏，则使用列的“display”属性，将其设置为“none”便不再显示此列，如果设置为“block”则会重新显示。



9.26 滚动的表格

【实例描述】

表格一般用来显示数据库中的信息，当数据库中内容过多时，可使用分页表格，也可以通过表格的滚动来显示数据。本例学习如何制作滚动的表格。

【实现代码】

```
<script type="text/javascript">
    marque(320,196,"icefable1","box1left")
    var scrollElem;
    var stopscroll;
    var stoptime;
    var preTop;
    var leftElem;
    var currentTop;
    var marqueesHeight;
//为表格添加事件
function marque(width,height,marqueName,marqueCName)
{
    try{
        marqueesHeight = height;
        stopscroll      = false;
        scrollElem = document.getElementById("mydiv");
        with(scrollElem){
            style.width      = width;
            style.height     = marqueesHeight;
            style.overflow   = 'hidden';
            noWrap           = true;
        }
        scrollElem.onmouseover = new Function('stopscroll = true');
        scrollElem.onmouseout  = new Function('stopscroll = false');
        preTop      = 0;
        currentTop  = 0;
        stoptime    = 0;
        leftElem = document.getElementById("mydiv");
        scrollElem.appendChild(leftElem.cloneNode(true));
        init_scrolltext();
    }catch(e) {}
}
//表格滚动的初始化
function init_scrolltext()
{
    scrollElem.scrollTop = 0;
    setInterval('scrollUp()', 18);
}
```

```
//向上滚动的方法
function scrollUp()
{
    if(stopscroll) return;
    currentTop += 1;
    if(currentTop == marqueesHeight+1) {
        stoptime += 1;
        currentTop -= 1;
        if(stoptime == (marqueesHeight)*1) {           //停顿时间
            currentTop = 0;
            stoptime = 0;
        }
    }else{
        preTop = scrollElem.scrollTop;
        scrollElem.scrollTop += 1;
        if(preTop == scrollElem.scrollTop){
            scrollElem.scrollTop = marqueesHeight;
            scrollElem.scrollTop += 1;
        }
    }
}
</Script>
```

需要在 body 中添加一个 id 为“mydiv”的层，其中带有表格，代码可参考随书光盘。

【运行效果】

表格滚动的效果如图 9-31 所示。

【难点剖析】

本例的重点是“scrollTop”属性。其在此例中表示表格的纵坐标位置，通过此值的递增来实现表格的滚动，图片和页面滚动同样是利用此属性。

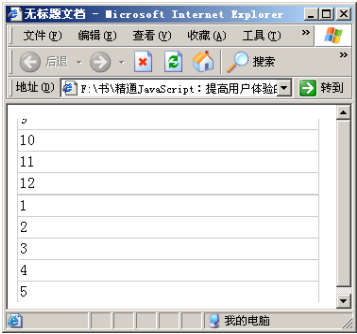


图 9-31 表格滚动的效果

9.27 交换表的行

【实例描述】

有时候表格需要进行排序，为了提高速度，可以在页面运行过程中使用 JavaScript 进行表格排序。本例只是学习如何通过交换表的行实现表格的排序。

【实现代码】

```
<script language="javascript">
function changerow()
{
    document.getElementById("mytbl").moveRow(2,1);
}
</script>
```

需要在 body 中，添加一个名为“mytbl”的表格，然后添加一个按钮，调用上述方法，代码如下所示：

```
<input id="Button1" type="button" value="交换行" onclick="changerow()" />
<table id="mytbl" width="300" height="50" border="0" cellspacing="2" cellpadding="0"
bgcolor="#FFb609">
  <tr>
    <td> 第一行第一列</td><td> 第一行第二列</td>
  </tr>
  <tr>
    <td> 第二行第一列</td><td> 第二行第二列</td>
  </tr>
</table>
```

【运行效果】

表格的初始运行效果如图 9-32 所示。表格换行后的效果如图 9-33 所示。

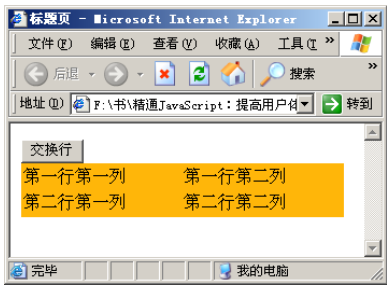


图 9-32 表格的初始运行效果

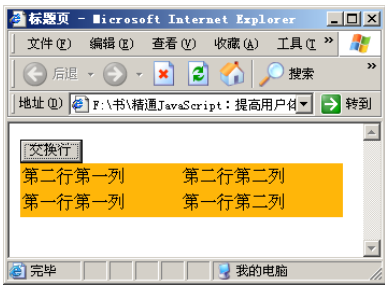


图 9-33 表格换行后的效果

【难点剖析】

本例的重点是“moveRow”方法，用来实现表格中行的交换。“moveRow”包含两个参数，第一个表示表格原来的行，第二个表示要交换到的目的行，注意参数 0 表示第一行，依次类推。

9.28 动态拖放表格的宽度

【实例描述】

表格的宽度一般在页面设计时被静态指定。本例学习如何在页面运行时，动态改变表格的宽度。

【实现代码】

```
<script language="javascript">
var dragenable=false;
var x;
var y;
var w;
var h;
```

```

var obj;
function init()
{
    x=event.clientX+document.body.scrollLeft;           //获取 X 坐标
    obj=event.srcElement;                               //获取鼠标触发的元素
    w=event.srcElement.offsetWidth;                     //对象的宽度
    obj.setCapture();                                    //设置鼠标消息
    if(x>event.srcElement.offsetLeft+w-5 && x<event.srcElement.offsetLeft+w){//鼠标移动到//
对象边界时
        dragenable=true;obj.style.cursor='e-resize';}    //改变鼠标的样式为左右拖动型
    }
    function drag()
    {
        if(event.clientX+document.body.scrollLeft>event.srcElement.offsetLeft+event.srcEl
ement.offsetWidth-5 && event.clientX+document.body.scrollLeft<event.srcElement.offsetLeft+
event.srcElement.offsetWidth)
            {event.srcElement.style.cursor='e-resize';}    //改变鼠标的样式为左右拖动型
        else
            event.srcElement.style.cursor='default';        //改变鼠标的样式为默认型
        if(dragenable==true) {
            if(event.clientX+document.body.scrollLeft-x+w>0) {
                var i=obj.cellIndex;
                var j;
                for(j=0;j<obj.parentNode.parentNode.rows.length;j++) { //更改表格的宽度
                    obj.parentNode.parentNode.rows[j].cells[i].width=event.clientX+ document.
body.scrollLeft-x+w;
                }
            }
            else {
                var i=obj.cellIndex;
                var j;
                for(j=0;j<obj.parentNode.parentNode.rows.length;j++) {
                    obj.parentNode.parentNode.rows[j].cells[i].width=1; //最小也要保持宽度为 1
                }
            }
        }
    }
    function end()                                     //结束更改
    {
        dragenable=false;
        obj.releaseCapture();                           //释放鼠标的捕获
        obj.style.cursor='default'; }                  //更改鼠标的样式为默认型
    }
}
</script>

```

【运行效果】

动态改变表格宽度的效果如图 9-34 所示。

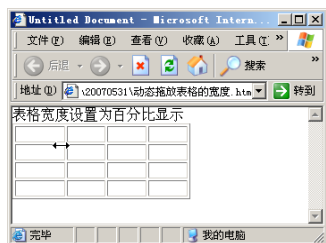


图 9-34 动态改变表格宽度的效果

【难点剖析】

本例的重点其实是鼠标坐标的获取。代码中使用“event.srcElement”获取鼠标操作的对象，然后使用“offsetWidth”获取该对象的宽度。当鼠标移动到表格边界时，再动态改变鼠标的样式。

9.29 可输入内容的表格

【实例描述】

默认的 HTML 表格只能输出数据，不能进行输入。本例用变通的方法，实现一个类似表格输入的效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script type="text/jscript">
    str="第一行|第一行|第二行|第二行";           // 默认单元格数据
    b=str.split("|")                               // 切割成数据数组
    for(var a in b)
        document.write("<input type='text' value="+b[a]+">") // 动态输出文本框
    </script>
</head>
<body>
</body>
</html>
```

【运行效果】

本例的运行效果如图 9-35 所示。

【难点剖析】

本例谈不上什么特殊技术，只是利用“input”输入框控件的边框，组成了一个类似“table”的效果。其实很多网站上页面的效果，不一定按照传统的思维模式去定义，变通一下也许会变得更简单，效果也看不出什么分别。

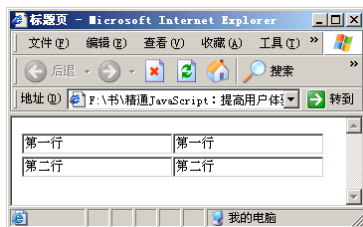


图 9-35 本例的运行效果

9.30 可以分级的表格隐藏

【实例描述】

分级的表格隐藏是指当用户单击第一行时，第一行后面的行都会隐藏；单击第二行时，则第二行下面的所有行会隐藏，依次类推。

【实现代码】

```

<html>
<head>
<meta http-equiv="Content-Language" content="zh-cn">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page</title>
<Script Language="javascript">
function trClick(c){
if(c==0){                                     //如果选择的是第一行
    if (document.all("a1").style.display=="")           //如果第二行是显示的
        document.all("a1").style.display="none";       //让第二行不显示
    else
        document.all("a1").style.display="";            //让第二行显示
    if (document.all("a1").style.display=="none")       //如果第二行不显示
        document.all("a11").style.display="none";      //第三行也不显示
    else
        document.all("a11").style.display="";          //第三行显示
}
else if(c==1){
    if (document.all("a11").style.display=="")          //如果第三行显示
        document.all("a11").style.display="none";      //第三行不显示
    else
        document.all("a11").style.display="";          //第三行显示
    if (document.all("a111").style.display=="")         //如果第四行显示
        document.all("a111").style.display="none";     //第四行不显示
    else
        document.all("a111").style.display="";         //第四行显示
}
}
else if(c==2){
    if (document.all("a111").style.display=="")         //如果第四行显示
        document.all("a111").style.display="none";     //第四行不显示
    else
        document.all("a111").style.display="";         //第四行显示
}
}
-->
</Script>
</head>

<body>
<div align="center">
<center>
<table border="0" cellpadding="0" cellspacing="0" width="200">
<tr id="a0" onclick="trClick(0)">
    <td height="24" bgcolor="#0000FF"><b><font color="#FFFFFF">第一行</font> </b></td>
</tr>
<tr id="a1" onclick="trClick(1)">

```



```
<td height="24" bgcolor="#FF0000"><font color="#FFFFFF">第二行</font></td>
</tr>
<tr id="all" onclick="trClick(2)">
  <td height="24" bgcolor="#FFFF00">第三行</td>
</tr>
<tr id="all1">
  <td height="24" bgcolor="#000000">第四行</td>
</tr>
</table>
</center>
</div>
</body>
</html>
```

【运行效果】

表格的原始效果如图 9-36 所示。单击第二行后的效果如图 9-37 所示。

【难点剖析】

本例的重点其实是一系列的“if...else if”语句。当单击第一行时，判断第一行后面所有行的显示状态，如果行显示属性“display”为空，则设置为“none”，表示不显示该行，否则显示该行。通过“else if”语句判断用户选择的是第几行，“c”的值通过单击事件中的参数传递。

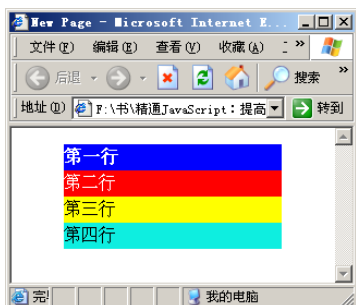


图 9-36 表格的原始效果

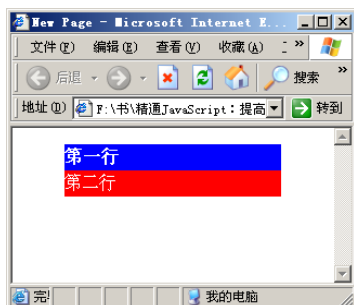


图 9-37 单击第二行后的效果

9.31 动态创建表格并实现分页

【实例描述】

表格的内容过多时，可以使用分页效果让用户可以更直观地了解表格内容。本例将动态生成一个带分页的表格。

【实现代码】

```
<html>
</head>
  <title>分页的动态表格</title>
  <style>div,table,td { text-align:center; }</style>
</head>
```

```
<body onload="ShowPage(1);">
  <div id="divTable"></div>
  <div id="divPages"></div>
</body>
</html>
<script language="javascript">
  var strData='医疗保健及个人用品类价格同比上涨 1.6%。其中西药价格下降 1.4%，中药材及中成药价格上涨 7.2%，医疗保健服务价格上涨 2.2%。';
  strData+='娱乐教育文化用品及服务类价格同比下降 1.2%。其中，学杂托幼费价格下降 0.7%。';
  //要显示的字符串

  var arrData=strData.match(/[\\u4e00-\\u9fa5]/ig);
  var numData=arrData.length;           //文本字符串的长度
  var numPages=Math.ceil(numData/30);    //返回除以 30 的上限
  function ShowPage(x)                  //显示指定页的内容
  {
    strData="";
    for (var i=(x-1)*30+1;i<=x*30;i++)
    {
      strData+="|<td>"+arrData[i]+"</td><td>"+arrData[++i]+"</td><td>"+arrData[++i]+"</td></tr>"
      //动态添加表格数据
    }
    strData='<table width="60%" border="5" cellpadding="0" cellspacing="0">'
    strData+strData+'</table>';
    document.getElementById("divTable").innerHTML=strData.replace (/<td>undefined\\</td>/ig,"");
    //在 div 内输出表格
  }
  for (var i=1;i<=numPages;i++)
  {
    var str='<button onclick="ShowPage(' +i+ ')">第'+i+'页</button>';
    //动态输出翻页按钮
    document.getElementById("divPages").innerHTML+=str;
    //在 div 内输出按钮
  }
</script>
|  |

```

【运行效果】

本例的运行效果如图 9-38 所示。

【难点剖析】

本例中固定每页显示的单元格数量为 30，然后使用“Math.ceil(numData/30)”判断共有多少页，最后使用 for 循环动态生成表格，并指定表格的内容。



图 9-38 本例的运行效果

9.32 删除表格指定行

【实例描述】

论坛中允许管理员删除一些不需要的主题，这时可以使用本例提供的删除指定行功能。删

除行后，下面的行自动上移。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function deleteRow (tableID, rowIndex)
{
    var table =document.all[tableID]                //获得当前表格
    table.deleteRow(rowIndex);                      //通过行索引删除行
}
</script>
</head>
<body>
<table id=mytable border=1><tr>
<td>第 1 行</td><td onclick="deleteRow('mytable',this.parentElement.rowIndex) ">删除当前
行</td>
</tr><tr><td>第 2 行</td><td onclick="deleteRow('mytable',this.parentElement. rowIndex)
"&>删除当前行</td>
</tr><tr><td>第 3 行</td><td onclick="deleteRow('mytable',this.parentElement. rowIndex)
"&>删除当前行</td>
</tr><tr><td>第 4 行</td><td onclick="deleteRow('mytable',this.parentElement. rowIndex)
"&>删除当前行</td>
</tr></table>
</body>
</html>
```

【运行效果】

页面初始运行效果如图 9-39 所示。删除第二行后的效果如图 9-40 所示。

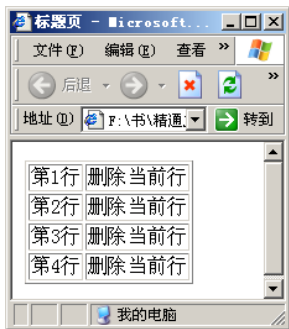


图 9-39 页面初始运行效果

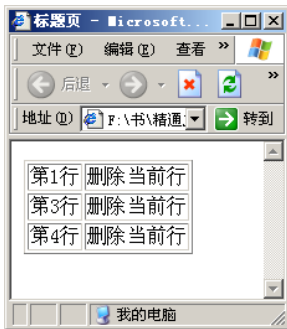


图 9-40 删除第二行后的效果

【难点剖析】

本例的难点是获取表格中某行的行号，以及删除表格行的方法。如果不将行号传给方法，在运行中是很难获取行号的，所以本例将行号作为参数传递给“deleteRow”方法。删除表格使用了表格对象自带的“deleteRow”方法，此方法的参数是要删除的行的行号。

9.33 设置表格的交替行颜色

【实例描述】

使用 ASP.NET 中的表格控件时，可以很方便地设计表格的交替行颜色。但 HTML 表格没有提供这种功能。本例将利用样式表实现这种交替变色的效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style>
tr{
    bgcolor:expression(this.bgColor=((this.rowIndex)%2==0 )? 'gray' : '#ffffcc');
}
</style>
</head>
<body>





```

【运行效果】

交替变换颜色的表格如图 9-41 所示。

【难点剖析】

本例的重点是设置交替变色的样式表。“rowIndex”用来判断表格的行索引，如果能被 2 整除，则颜色设置为“gray”，否则设置为“#ffffcc”。注意代码中改变样式的语法。

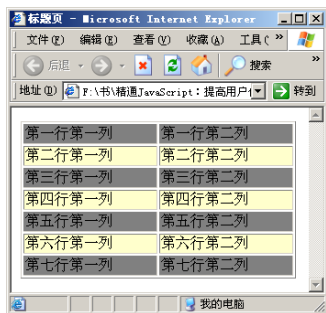


图 9-41 交替变换颜色的表格

9.34 双击单元格变为可编辑

【实例描述】

ASP.NET 中提供了多个表格编辑控件，可通过双击实现单元格的编辑。本例介绍一种方法，可实现双击单元格编辑 HTML 表格的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script>
function editCell(obj){
    if(obj.innerText==""){
        obj.innerHTML="<input value='' onblur='update(this.value)'">;
        //插入输入框，失去焦点时更新
    }else{
        obj.innerHTML="<input value="+obj.innerText+" onblur='update(this. value)'">;
        //插入文本框，且指定内容
    }
}
function update(txt){
document.getElementById("Td2").innerText=txt;    //文本框失去焦点时，需要更新表格的内容
}
</script>
</head>
<body>
<table border="1"><tr><td id="Td2" onDblClick="editCell(this)">第一行第一列</td>
</tr></table></body>
</html>
```

【运行效果】

默认生成的表格如图 9-42 所示。双击单元格后的表格如图 9-43 所示。

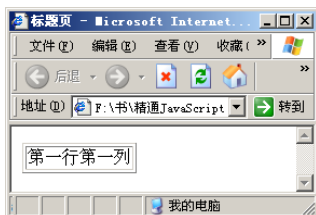


图 9-42 默认生成的表格

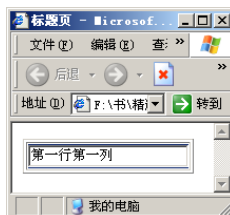


图 9-43 双击单元格后的表格

【难点剖析】

本例的重点是在表格内动态添加“input”控件。因为添加的代码中有 HTML 语句，所以必须使用“innerHTML”属性实现“input”控件的动态插入。如果表格中已经有值，需要设置新“input”控件的“value”属性。当文本框被改变且失去焦点时，需要将文本框的内容更新回表格。本例通过“update”方法实现这种更新。

9.35 鼠标经过表格时列变色

【实例描述】

很多例子是讲如何在鼠标移过时，改变当前行的颜色。本例学习如何在鼠标移过时，改变

当前列的颜色。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<table width=500 cellpadding=0 rules=groups border
onmouseout="setColor0(event.fromElement)"
onmouseover="setColor1(event.srcElement)">
<col><col><col><col>
<tr><td>第一列</td><td>第二列</td><td>第三列</td><td>第四列</td></tr>
<tr><td>第一列</td><td>第二列</td><td>第三列</td><td>第四列</td></tr>
<tr><td>第一列</td><td>第二列</td><td>第三列</td><td>第四列</td></tr>
<tr><td>第一列</td><td>第二列</td><td>第三列</td><td>第四列</td></tr>
</table><script>
var cols=document.getElementsByTagName("table")[0].children[0].children;
//获取所有列

function setColor0(sender)
{
    if(sender.tagName=="TD")
        cols[sender.cellIndex].style.backgroundColor="";
//鼠标移走时，取消颜色
}
function setColor1(sender)
{
    if(sender.tagName=="TD")
        cols[sender.cellIndex].style.backgroundColor="gray"; //鼠标移过来时，改变颜色
}
</script>
</body>
</html>
```

【运行效果】

鼠标移过列时的效果如图 9-44 所示。

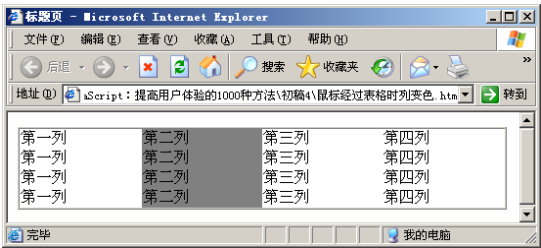


图 9-44 鼠标移过列时的效果

【难点剖析】

本例的重点是获取当前列。在传递事件参数时，使用“event.fromElement”传递当前操作



的对象。使用“sender.tagName”判断当前操作对象是否是列，如果是，则将当前所有列的背景色都改变。

9.36 鼠标选择表格中的多行

【实例描述】

可通过拖曳的形式实现选择表格的多行。本例通过改变行的背景色，演示选择表格多行的效果。

【实现代码】

```
<html>
<head>
<title> 鼠标选择多行</title>
</head>
<script language="JavaScript">
var curObj= null;
var isTrue =false;
function setSelectedBgColor()//设置选定行的背景色
{
isTrue =true;
if(window.event.srcElement.tagName=='TD'){           //如果选中的是单元格
    curObj=window.event.srcElement.parentElement;    //获取其父级节点——表格行
    curObj.style.background='#ff9999';                //设置背景色
}
}
</script>
<body>
<table border="1" width=220 height=220 onmousemove= "if(isTrue) setSelectedBgColor();"
onmousedown="setSelectedBgColor();" onmouseup="isTrue =false;">
<tr>
<td>单元格</td>
<td>单元格</td>
<td>单元格</td>
</tr>
<tr>
<td>单元格</td>
<td>单元格</td>
<td>单元格</td>
</tr>
<tr>
<td>单元格</td>
<td>单元格</td>
<td>单元格</td>
</tr>
```



```
<td>单元格</td>
<td>单元格</td>
<td>单元格</td>
</tr>
</table>
</body>
</html>
```

【运行效果】

选中多行后的效果如图 9-45 所示。

【难点剖析】

首先判断用户选择的是否是单元格，如果是，则获取单元格的父级标签“tr”。最后通过“background”属性改变行的背景色。

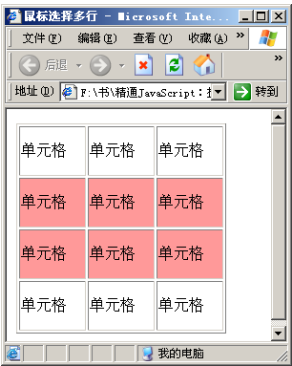


图 9-45 选中多行后的效果

9.37 使用 JavaScript 向表格中写入数据

【实例描述】

本例向表格中写入数据，类似于在表格中动态插入单元行。不同的是在插入的同时指定单元格的内容，并通过一个方法提高代码的复用性。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="JavaScript">
//---变量-----释义--
//idNumber -----音乐编号
//songName-----音乐名称
//mCompany-----制作公司
//mSinger-----演唱歌手
//mPlace-----发行区域
//mKind-----音乐类型
//mTime-----加入日期
//mSize-----音乐大小
//mType-----音乐格式
//mUrl-----音乐地址
//mHits-----点播次数

function createTR(idNumber,songName,mPlace,mCompany,mSinger)
{
    document.writeln("<TR onmouseover=\"this.style.backgroundColor= '#FFccdd';this.
style.color='BLUE'\" onmouseout=\"this.style.backgroundColor=''; this.style.color='\">");
    document.writeln("<TD height=10>"+idNumber+"</TD>");
    document.writeln("<TD height=10><a href='#\">"+songName+"</a></TD>");
```



```
document.writeln("<TD height=10>"+mPlace+"</TD>");
document.writeln("<TD height=10>"+mCompany+"</TD>");
document.writeln("<TD height=10>"+mSinger+"</TD>");
document.writeln("</TR>");
}
</script>
<TABLE width="90%" bgcolor="#BCBCBC" cellSpacing=1 cellPadding=0 align= center border=1>
<TR>
<Th width="20%">音乐编号</Th>
<Th width="20%">音乐名称</Th>
<Th width="20%">发行区域</Th>
<Th width="20%">所属公司</Th>
<Th width="20%">演唱歌手</Th>
</TR>
<TR>
<script>
createTR("编号 1","歌曲名称 1","中国大陆","公司 1","歌手 1");
createTR("编号 2","歌曲名称 2","中国香港","公司 2","歌手 2");
createTR("编号 3","歌曲名称 3","中国台湾","公司 3","歌手 3");
createTR("编号 4","歌曲名称 4","中国澳门","公司 4","歌手 4");
createTR("编号 5","歌曲名称 5","中国大陆","公司 5","歌手 5");
</script>
</TR>
</table></body>
</html>
```



图 9-46 本例的运行效果
“document.write”。

【运行效果】

本例的运行效果如图 9-46 所示。

【难点剖析】

本例的效果类似于一个音乐选播器,代码中自定义了一个“createTR”方法,可实现动态表格行的插入。插入的内容由指定的参数获取。注意如果要在文档中输出一行,使用“document.writeln”,而不是

9.38 类 C# GridView 的编辑效果(一)

【实例描述】

在 C#中,单击 GridView 的编辑按钮可实现表格的编辑。本例通过一个表格和链接实现类似的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>test</title>
```

```

</head>
<body>
<table>
  <tr><td id="td1">张三</td>
  <td><a href="javascript:edit()">编辑</a></td></tr>
</table>
<script language="javascript">
function edit()
{
  var cell = document.getElementById('td1');           //获取表格
  var v = cell.innerHTML;                               //获取表格的内容
  cell.innerHTML = '<input type="text" name="text1" id="text1">'; //动态添加文本框
  document.getElementById('text1').value = v;           //设置文本框的值
}
</script>
</body>
</html>

```

【运行效果】

类 C# GridView 的编辑效果如图 9-47 所示。

【难点剖析】

本例的难点是获取需要编辑的表格元素，并在其中动态添加一个输入框。注意获取表格元素的值可使用“innerHTML”属性。

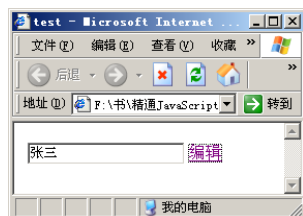


图 9-47 类 C# GridView 的编辑效果

9.39 类 C# GridView 的编辑效果（二）

【实例描述】

表格内可直接输入文本，输入完毕后，还可以将内容自动保存在单元格内。本例实现的就是表格的这种编辑功能。

【实现代码】

```

<script language="javascript">
oldObj="";
var newInput=document.createElement("input");           //动态创建输入框
newInput.type="text";                                   //输入框类型
function setEdit(e){                                     //设置编辑时的状态
  var tdObj = e.srcElement? e.srcElement : (e.target ? e.target : e); //单击的对象

  var obj;
  if(tdObj.tagName && tdObj.tagName=="TD"){              //判断是否是单元格
    if(oldObj!=""){
      var tobj = document.getElementById('tmpText');    //判断是否已经存在输入框
      oldObj.removeChild(tobj);                          //移除已经存在的输入框
      if(newInput.vlaue=="")                             //初始化输入框的值
        oldObj.innerHTML="&nbsp;";
      else
        oldObj.innerHTML=newInput.value;                //输入框的内容等于单元格的内容
    }
  }
}

```



```
    }

    obj=tdObj;
    oldObj=obj;
    newInput.width=obj.offsetWidth;           //输入框的高度和宽度
    newInput.height=obj.offsetHeight;

    newInput.id="tmpText";
    newInput.value=obj.innerHTML;
    obj.innerHTML="";
    obj.appendChild(newInput);                //将输入框添加到单元格内
    newInput.focus();                          //输入框获得焦点
}
tdObj = obj = tobj = null;
}
function checkAdd(e){
    if(e && e.keyCode == 13){
        var obj = e.srcElement? e.srcElement : e.target; //获得单击对象
        var tbl = obj.parentNode.parentNode;           //单击对象的祖父节点
        if(oldObj!=""){
            var tobj = document.getElementById('tmpText'); //获取输入框
            oldObj.removeChild(tobj);                     //移除旧输入框
            if(newInput.vlaue=="")                        //单元格的初始值
                oldObj.innerHTML="&nbsp;";
            else
                oldObj.innerHTML=newInput.value;          //单元格的内容等于输入框的内容
            var oldObj2 = oldObj;
            oldObj2 = '';
        }
        if(tbl.tagName && tbl.tagName == 'TR'){          //如果是单元行
            t2 = tbl.cloneNode(true);                    //克隆表格
            tbl.parentNode.insertBefore(t2,tbl);          //插入行
        }
        setEdit(oldObj2);                                //开始编辑
    }
    obj = tbl = tobj = t2 = oldObj2 = null;
}
</script>
```

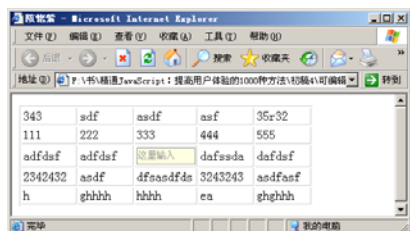


图 9-48 表格的编辑效果

【运行效果】

表格的编辑效果如图 9-48 所示。

【难点剖析】

本例的原理是当用户单击单元格时,自动生成一个用于输入的“input”控件,此控件的高度和宽度与单元格相同,其值也等于单元格的内容。当鼠标离开单元格时,隐藏此输入框并设置单元格的内容等于输入框的值。

第 10 章

单选按钮和复选框

本章导读

单选按钮表示一组数据只能选择其中的一项。复选框表示可以选择一组数据中的多项。本章主要介绍复选框的一些常用技巧，如全选、复选、分组选择等。

10.1 选择了哪一个单选按钮

【实例描述】

单选按钮是一组类似的数据，但用户只能选择其中的一个值。本例学习如何判断用户选择了哪个值。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
function checkradio()
{
    var parms=document.getElementsByName("radio1");    //获取所有的单选按钮
    var i;
    for( i=0;i<parms.length;i++)                        //遍历单选按钮
    {
        if(parms[i].checked)                            //如果选择了此单选按钮
            alert("您选择了"+ parms[i].value);          //提示用户的选择
    }
}
</script>
</head>
<body>
<input name="radio1" type="radio" value="体育运动" checked=checked />体育运动
<input name="radio1" type="radio" value="旅游休闲"/>旅游休闲
<input type="button" name="btn1" value="检查" onclick="checkradio()" />
</body>
</html>
```

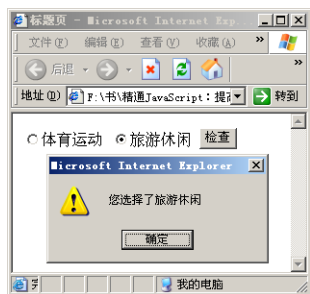


图 10-1 单选按钮选择后的效果

【运行效果】

单选按钮选择后的效果如图 10-1 所示。

【难点剖析】

本例重点是如何获取页面中的所有单选按钮，然后判断单选按钮的“checked”属性。“checked”属性表示单选按钮是否被选中。获取单选按钮的值使用“value”属性。

10.2 单击文字实现单选按钮的选定

【实例描述】

在网页的基本控件中，单选按钮控件是不包含文字的，在页面单选按钮后面的文字其实只是页面的布局，两者之间实际上没有任何关系。但在用户看来，两者是一体的。本例的目的就

是将两者设计为一体，实现选择文字的同时选中单选按钮。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<br>
<input type="radio" name="myLike" value="Like1" id="Like1"><label for="Like1"> 体育运动:NBA</label> <br>
<input type="radio" name="myLike" value="Like2" id="Like2"><label for="Like2"> 休闲运动:Golf</label>
</body>
</html>
```

【运行效果】

单击文字后的效果如图 10-2 所示。

【难点剖析】

本例的重点是 label 元素的“for”属性。此属性用来设置 label 是为谁服务的，属性的值必须是被服务控件的 id。

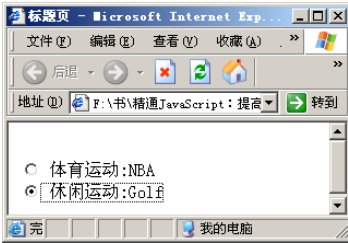


图 10-2 单击文字后的效果

10.3 被选中的复选框求和

【实例描述】

在一些购物网站的购物篮设计中，要求在用户选中商品后自动计算商品的总价格。本例通过一个简单的复选框，实现这种计算总和的效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function dataCal(){
var sum=0;
var obj=document.getElementsByName("items"); //取得页面所有的 items 复选框对象
for(var i=0;i<obj.length;i++)
{
if(!obj[i].checked)
continue; //如果没有选中，则执行下一次
sum+=parseFloat(obj[i].value); //如果被选中的话，则累加求和
}
alert("总和为:"+sum);
}
```



```
</script>
</head>
<body>
<input type="checkbox" value="11" name="items">11<br/>
<input type="checkbox" value="22" name="items">22<br/>
<input type="checkbox" value="33" name="items">33<br/>
<input type="checkbox" value="44" name="items">44<br/>
<input type="button" value="求和" onclick="dataCal()"></body>
</html>a
```

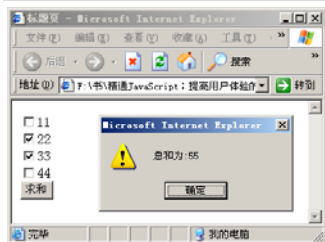


图 10-3 本例的运行效果

【运行效果】

本例的运行效果如图 10-3 所示。

【难点剖析】

本例的重点是复选框控件的“value”值。在用户选中复选框后，获取复选框的“value”值，并使用“parseFloat”方法进行类型转换。

10.4 复选框组选

【实例描述】

复选框的功能是用来多选，当复选框有多组数据时，如何实现同时选中某一组数据呢？本例学习这种效果的制作过程和原理。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
function sts()
{
}
</script>
</head>
<body>
<script language="JavaScript">
function selectGroup(obj)
{
    var chkObj = obj.id; //获取对象的 id
    var chkValue=obj.checked; //判断当前对象是否选中
    var chkArr = mySpan.getElementsByTagName("input"); //获取所有的复选框
    for(var i=0; i<chkArr.length; i++) //遍历所有的复选框
    {
        if(chkArr[i].id==chkObj) //判断 id 是否相等
        {
            chkArr[i].checked=chkValue //如果相等则选中
        }
    }
}
```



```
</script>
<span id="mySpan">
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="A001" value="1">A001
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="B001" value=
"3">>B001
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="A001" value=
"444">>A001<br />
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="B001" value=
"23">>B001
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="B001" value=
"11">>B001
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="C001" value=
"15">>C001<br />
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="C001" value=
"17">>C001
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="C001" value=
"10">>C001
  <input type="checkbox" onclick="selectGroup(this)" name="ID[]" id="C001" value=
"121">>C001
</span>
</body>
</html>
```

【运行效果】

分组选中后的效果如图 10-4 所示。

【难点剖析】

本例为同组的复选框设置了相同的 id，这在 ASP.NET 中一般是不允许的。当用户选中某个复选框时，通过“obj.id”获取其 id，然后遍历页面中的所有复选框，并指定相同 id 的复选框状态相同。



图 10-4 分组选中后的效果

10.5 复选框分组全选

【实例描述】

可以通过遍历窗体中的复选框实现复选框的全选。如果窗体中的复选框进行了分组，那该如何实现某一组复选框的全选呢？

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<form id="form1" name="form1" method="post" action="">
  <table width="120" border="0">
```



```
<tr>
  <td>教师组/学生组</td>
</tr>
<tr>
  <td><input name="check" type="checkbox" id="check" value="yes" />
  <input name="nocheck" type="checkbox" id="nocheck" value="no" /></td>
</tr>
<tr>
  <td><input name="check" type="checkbox" id="Checkbox1" value="yes" />
  <input name="nocheck" type="checkbox" id="Checkbox2" value="no" /></td>
</tr>
<tr>
  <td><input name="check" type="checkbox" id="Checkbox3" value="yes" />
  <input name="nocheck" type="checkbox" id="Checkbox4" value="no" /></td>
</tr>
<tr>
  <td><input name="check" type="checkbox" id="Checkbox5" value="yes" />
  <input name="nocheck" type="checkbox" id="Checkbox6" value="no" /></td>
</tr>
<tr>
  <td><input name="check" type="checkbox" id="Checkbox7" value="yes" />
  <input name="nocheck" type="checkbox" id="Checkbox8" value="no" /></td>
</tr>
<tr>
  <td><input name="check" type="checkbox" id="Checkbox9" value="yes" />
  <input name="nocheck" type="checkbox" id="Checkbox10" value="no" /></td>
</tr>
<tr>
  <td><input name="check" type="checkbox" id="Checkbox11" value="yes" />
  <input name="nocheck" type="checkbox" id="Checkbox12" value="no" /></td>
</tr>
<tr>
  <td>全选教师
  <input name="checkall" type="checkbox" id="checkall" value="checkbox" onclick=
"change(document.getElementsByName('check'), this.checked)" /></td>
</tr>
<tr>
  <td>全选学生
  <input name="nocheckall" type="checkbox" id="nocheckall" value="checkbox" onclick=
"change(document.getElementsByName('nocheck'), this.checked)" /></td>
</tr>
</table>
</form>
<script type="text/javascript">
var change = function (chkArray, val)
{
  for (var i = 0 ; i < chkArray.length ; i ++) //遍历指定组中的所有项
    chkArray[i].checked = val;                //设置项为指定的值——是否选中
}
```

```
}  
</script>  
</body>  
</html>
```

【运行效果】

本例的选择效果如图 10-5 所示。

【难点剖析】

本例的重点是“getElementsByName”方法，其可以获取指定“name”属性的所有标签，这样就可以获取某一组复选框。然后使用“change”方法，更改复选框的选中状态即可实现本例效果。

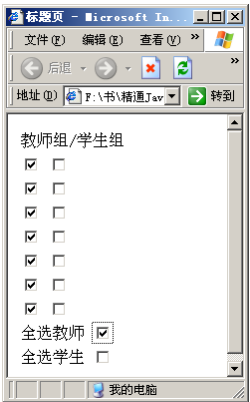


图 10-5 本例的选择效果

10.6 复选框和文本框的联动效果

【实例描述】

所谓联动效果就是一个控件不能用时，另一个控件也不能用。本例学习复选框和文本框的联动效果。当复选框没被选中时，文本框处于不可用状态。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
<script language="javascript">  
function chk(obj,txtIndex)  
{  
    var tmpInput = document.getElementsByName("textfield")[txtIndex]; //获取指定的文本框  
    obj.checked?tmpInput.disabled=false:tmpInput.disabled=true;  
    //判断文本框是否处于选中状态  
}  
</script>  
</head>  
<body>  
<form id="form1" name="form1" method="post" action="">  
    <input name="che" type="checkbox" value="" onclick="chk(this,'0')"/>  
    <input type="text" name="textfield" disabled=true /><BR/>  
    <input name="che" type="checkbox" value="" onclick="chk(this,'1')"/>  
    <input type="text" name="textfield" disabled=true /><BR/>  
    <input name="che" type="checkbox" value="" onclick="chk(this,'2')"/>  
    <input type="text" name="textfield" disabled=true /><BR/>  
</form>  
</body>  
</html>
```

【运行效果】

本例的运行效果如图 10-6 所示。

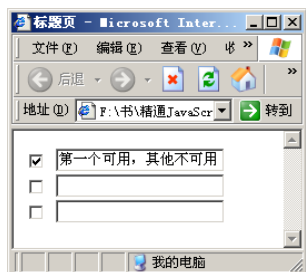


图 10-6 本例的运行效果

【难点剖析】

设置文本框是否可用的属性是“disabled”，其值为“true”表示不可用，为“false”表示可用。在“chk”方法中，使用“obj”获取页面中所有的文本框，通过第二个参数“txtIndex”判断当前应该对应哪个文本框。三元运算符“表达式？值 1：值 2”用来表示当复选框选中时文本框设置为可用，否则为不可用。

10.7 单击任意单元格都能自动选中复选框

【实例描述】

为了方便用户操作，当用户要选中复选框时，不需要单击复选框控件，而只要双击复选框控件所在行中任意单元格即可。本例学习如何实现这种效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="javascript" type="text/javascript">
function chkSelect(){
    var tr = event.srcElement.parentElement;    //获取当前操作对象的父级对象
    tr.cells[0].children[0].checked=true;        //选中父对象的第一个元素即复选框
}
</script>
<table width="285" border="1">
    <tr onclick='chkSelect();'>
        <td><input type="checkbox" name="checkbox" value="checkbox">
    </td>
        <td>第一行第一列</td>
        <td>第一行第二列</td>
    </tr>
    <tr onclick='chkSelect();'>
        <td><input type="checkbox" name="checkbox2" value="checkbox"></td>
        <td>第二行第一列</td>
        <td>第二行第二列</td>
    </tr>
</table></body>
</html>
```

【运行效果】

双击单元格后的效果如图 10-7 所示。

【难点剖析】

注意本例的双击事件是绑定在表格的行上，当用户双击某行时，首先通过“event.srcElement”属性获取双击的单元格，然后使用 DOM 对象的“parentElement”属性获取单元格的父级对象“tr”。因为复选框控件位于行的第一列位置，所以可以使用“children[0]”获取行中的复选框，最后设置其“checked”属性即可。

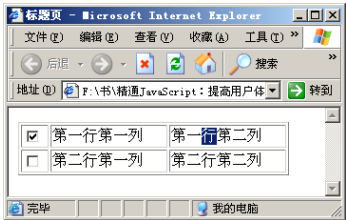


图 10-7 双击单元格后的效果

10.8 调用复选框后面的文字

【实例描述】

在 HTML 中，复选框和复选框后面的文字并不是一体的，不能通过复选框的“value”属性获取后面的文字。本例学习如何调用复选框后面的文字。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<form name="frm">
<INPUT id=checkList type=checkbox name=checkList onclick="window.alert (nextSibling.
innerText);"><LABEL for=checkList>这是后面的文字</LABEL>
</form>
</body>
</html>
```

【运行效果】

调用复选框后文字的效果如图 10-8 所示。

【难点剖析】

本例在复选框后面使用 label 标签显示文字。“onclick”事件中使用了“nextSibling”属性，此属性一般用于 DOM（文档对象模型）中，表示当前元素的下一个元素。此例中的下一个元素便是“Label”，用此属性便实现了对复选框后面文字的调用。

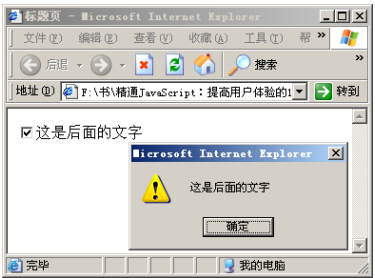


图 10-8 调用复选框后文字的效果

10.9 两组复选框互斥问题

【实例描述】

有时候可以将复选框分为两组，选择了 A 组就不能选择 B 组。如在一些运动项目中，运动



员选择了 200 米，可能就不能选择 200 米接力。本例通过两组复选框，学习如何实现两组选项之间的互斥。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function sel1(obj){
    if(obj.checked){                                     //如果当前项被选中了
        document.all.chk2.checked=false               //则不能选择第二组
        document.all.sel1.selectedIndex=4              //指定下拉框中选择的项
    }
}
function sel2(obj){
    if(obj.checked){                                     //如果当前项被选中了
        document.all.chk1.checked=false                 //则不能选择第一组
        document.all.sel1.selectedIndex=0               //指定下拉框中选择的项
    }
}
</script>
</head>
<body>
<p>A 组
    <input type="checkbox" name="checkbox" value="checkbox" id="chk1" onClick="sel1(this)">
</p>
<p>B 组
    <input type="checkbox" name="checkbox2" value="checkbox" id="chk2" onClick="sel2(this)">
</p>
<p>C
    <select name="select" id="sel1">
    <option selected="selected">个人</option>
    <option>200 米</option>
    <option>400 米</option>
    <option>800 米</option>
    <option>200 米接力</option>
    </select>
</p>
</body>
</html>
```

【运行效果】

本例的运行效果如图 10-9 所示。

【难点剖析】

本例的重点是如何实现两组复选框的互斥。“checked”属性用来设置复选框是否选中，其值如果为“true”则表示选中，为“false”表示未选中。当改变“A 组”复选框选中状态的时候，

需要先判断其是否被选中，如果被选中则设置“B 组”复选框的“checked”属性为“false”，否则设置为“true”。

10.10 使用复选框控制文本框

【实例描述】

很多表格的第一列是复选框，允许用户选择某行进行操作。本例学习如何使用复选框，控制某行的文本框。

【实现代码】

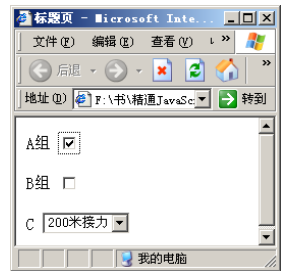


图 10-9 本例的运行效果

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>Untitled Document</title>
<script language="javascript">
function chkOper(chk)
{
    var inputs = chk.parentNode.parentNode.cells[1].getElementsByTagName("input");
    //通过父级节点获取输入框
    var status = chk.checked;
    //判断复选框是否选中的变量
    for(var i=0,j=inputs.length;i<j;i++)
    //遍历每个 input 控件
        inputs[i].disabled = status;
    //通过复选框的值设置每个 input 控件的可用性
}
</script>
</head>

<body>
<table width="500" border="1" cellspacing="1" cellpadding="1">
<tr>
<td width="40px" align="center">
<input type="checkbox" name="checkbox" id="" onclick="chkOper(this)"/></td>
<td width="230px">
<input type="text" name="textfield" id="Text1" />
</td>
<td width="230px">第一行</td>
</tr>
<tr>
<td align="center"><input type="checkbox" name="checkbox" id="Checkbox1" onclick=
"chkOper(this)"/></td>
<td><input type="text" name="textfield2" id="textfield2" /></td>
<td>第二行</td>
</tr>
<tr>
<td align="center"><input type="checkbox" name="checkbox" id="Checkbox2" / onclick=
"chkOper(this)"/></td>
<td><input type="text" name="textfield3" id="textfield3" /></td>
<td>第三行</td>
</tr>
</table>
</body>
</html>
```

```
</tr>
</table>
</body>
</html>
```

【运行效果】

本例的运行效果如图 10-10 所示。

【难点剖析】

本例的重点是 DOM 中节点的应用。首先获取复选框控件，然后使用“parentNode”获取父节点。获取到表格控件后，再使用 table 的列属性“cells”找到 input 控件，最后使用“disabled”属性控制文本框是否能输入。



图 10-10 本例的运行效果

10.11 选中表格行前的复选框则行变色

【实例描述】

ASP.NET 提供一个网格控件，当用户选择某行时，整行的颜色都会发生变化，但 HTML 中的表格不具有此特效。本例学习如何通过 JavaScript 变通的手法，实现普通表格整行颜色变化的特效。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<table width="50%" border="0" cellspacing="0" cellpadding="0" align="center">
<tr>
<td bgcolor="#CC6655"><input type="checkbox" name="checkbox" value="checkbox"
onClick="if(this.checked){this.parentNode.parentNode.style. background='#cc5566'}else
{this.parentNode.parentNode.style.background=' '}">
全选</td>
</tr>
<tr>
<td><table id="tab1" width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
```



```

        <td><input type="checkbox" name="checkbox" value="checkbox" onclick=" if(this.
checked){this.parentNode.parentNode.style.background='#cc5566'}else{this.parentNode.paren
tNode.style.background=''}">
        第一行</td><td>第二列</td>
    </tr>
    <tr>
        <td><input type="checkbox" name="checkbox" value="checkbox" onclick="if(this.
checked){this.parentNode.parentNode.style.background='#cc5566'}else{this.parentNode.paren
tNode.style.background=''}">
        第二行</td><td>第二列</td>
    </tr>
</table></td>
</tr>
</table></body>
</html>

```

【运行效果】

选中复选框后的表格行效果如图 10-11 所示。

【难点剖析】

本例在表格的第一列中添加了“checkbox”标签，实现复选框效果。当用户选中复选框时，触发其“onclick”事件。“parentNode”表示当前选中元素的父节点，当前元素是“checkbox”，其父节点是“td”标签，因为要求全行都选中，所以还必须再选择上一级父节点“tr”标签，本例使用了“this.parentNode.parentNode”。修改表格行的背景色使用“background”属性。



图 10-11 选中复选框后的表格行

10.12 用 JavaScript 生成面包屑导航

【实例描述】

面包屑导航是一种不同于网站导航的常用页面导航方式。本例学习如何通过一系列复选框实现面包屑导航的效果。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type="checkbox" name="ck" value="子目录 A">子目录 A
<input type="checkbox" name="ck" value="子目录 B">子目录 B
<input type="checkbox" name="ck" value="子目录 C">子目录 C
<input type="checkbox" name="ck" value="子目录 D">子目录 D
<input type="checkbox" name="ck" value="子目录 E">子目录 E
<input type="button" value="生成" onclick="create();"><br>

```



```
<div id="子目录 E" style="color:gray"></div>
<script>
function create()
{
    var arr = document.getElementsByName("ck"); //获取所有的复选框元素，保存在数组中
    var newarr = [];
    for(var i=0;i<arr.length;i++){                //遍历所有的复选框
        if(arr[i].checked){                        //判断某项是否被选中
            newarr.push(arr[i].value);            //添加到新数组中
        }
    }
    子目录 E.innerHTML = newarr.join("->");        //生成面包屑效果
}
</script>
</body>
</html>
```

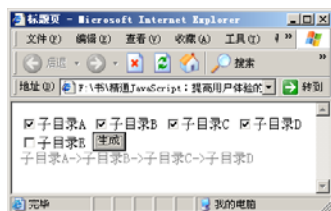


图 10-12 生成的面包屑效果

【运行效果】

生成的面包屑效果如图 10-12 所示。

【难点剖析】

本例通过两个数组实现了内容的选择，然后通过“->”连接符号实现了面包屑效果。其中数组的“push”方法，用来将某个值添加到数组中。

10.13 复选框的反选

【实例描述】

当用户选择的复选框个数太多，而且超过复选框总数的 2/3 时，为了简化用户操作，可以让用户先选择不需要的项，然后使用“反选”按钮，实现正确的选择。操作系统的资源管理器就提供了这种功能，用来实现文件的反选。本例介绍如何实现复选框的反选。

【实现代码】

```
<HTML>
<HEAD>
<title>反选</title>
<script language=javascript>
function selectOther(chkObj)
{
    for(var i = 0;i<chkObj.elements.length;i++) //遍历窗体中的元素
    if(chkObj.elements[i].type == "checkbox" )      //判断元素类型是否是复选框
    {
        if(!chkObj.elements[i].checked)         //如果复选框未选中
            chkObj.elements[i].checked = true;   //改为选中状态
        else
    }
```

```
        chkObj.elements[i].checked = false;           //改为未选中状态
    }
}
</script>
</HEAD>
<BODY>
<form name="form1">
    <input id="Checkbox1" name="chk" type="checkbox" />旅游<br />
    <input id="Checkbox2" name="chk" type="checkbox" />体育<br />
    <input id="Checkbox3" name="chk" type="checkbox" />音乐<br />
    <input id="Button1" type="button" value="反选" onclick="selectOther (document. form1)" />
</form>
</BODY>
</HTML>
```

【运行效果】

初始选择的效果如图 10-13 所示。反选后的效果如图 10-14 所示。

【难点剖析】

本例的重点是如何遍历窗体中所有的复选框控件。本例使用 for 循环语句，通过“type”属性逐个判断窗体中每个元素的类型，如果类型为“checkbox”，则判断其是否被选中，然后设置复选框为反选状态即可。

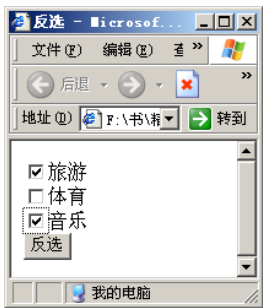


图 10-13 初始选择的效果



图 10-14 反选后的效果

10.14 复选框全选（一）

【实例描述】

在浏览网页时经常需要全选所有复选框，为了简化用户操作，可以提供“全选”按钮，用来选中页面中所有的复选框，或者取消所有的选择。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>全选</title>
<script type="text/javascript">
```



```
function SelectAll()
{
    oEl = event.srcElement;           // 获取当前单击的元素
    for(i = 0;i < document.all.length; i++)
    {
        // 遍历所有的 checkbox
        if(document.all(i).id.indexOf("Checkbox") != -1)
        {
            if(oEl.checked)           // 如果选择了全选
                document.all(i).checked = true;    // 全选
            else
                document.all(i).checked = false;    // 全不选
        }
    }
}
</script>
</head>
<body>

    <input id="Checkbox1" type="checkbox" value="a" /><label >争取举办奥运会的城市
</label><br />
    <input id="Checkbox2" type="checkbox" value="b"/><label >举办过奥运会的城市
</label><br />
    <input id="Checkbox3" type="checkbox" value="c" /><label >成功举办奥运会的城市
</label><br />
    <input id="Checkbox4" type="checkbox" value="d" /><label >亚洲举办过奥运会的城市
</label><br />
    <input id="Checkbox5" type="checkbox" onclick="SelectAll()" />全选或全不选
</body>
</html>
```

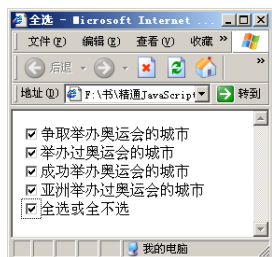


图 10-15 全选复选框的效果

【运行效果】

全选复选框的效果如图 10-15 所示。

【难点剖析】

本例的重点是如何获取页面中所有的“checkbox”。此处使用的是方法是遍历页面中所有的元素，然后判断元素的 id 中是否包含“checkbox”字符串，如果包含则说明此元素是复选框。最后设置复选框的“checked”属性值为“true”（选中）或“false”（不选）。

10.15 复选框全选（二）

【实例描述】

复选框的全选功能通过服务器端开发语言可以很方便地实现，但有时为了提高页面的访问速度，需要使用 JavaScript 完成。本例学习实现全选的 JavaScript 方法。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function checkAll(str)
{
    var a = document.getElementsByName(str);           //获取所有复选框
    var n = a.length;                                   //获取复选框的个数
    for (var i=0; i<n; i++)
        a[i].checked = window.event.srcElement.checked; //通过单击的按钮判断是选中还是未选
}
</script>
</head>
<body>
<input type=checkbox name=All onclick="checkAll('ck')">全选<br/>
<input type=checkbox name=ck >体育<br/>
<input type=checkbox name=ck >旅游<br/>
<input type=checkbox name=ck >饮食<br/>
<input type=checkbox name=ck >影视<br/>
<input type=checkbox name=ck >音乐<br/><br/></body>
</html>

```

【运行效果】

全部选择的效果如图 10-16 所示。

【难点剖析】

本例的重点是如何获取页面中所有的复选框控件。使用“document.getElementsByName”可以获取页面中所有的复选框控件，然后使用“window.event.srcElement.checked”判断是选中还是未选。



图 10-16 全部选择的效果

10.16 获取复选框的选择项

【实例描述】

复选框由多个 checkbox 构成。本例学习如何判断用户的选择，并获取用户的选择值。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
function mySelect()
{
    var i=0;

```



```
var j=0;
var selectObj=document.getElementsByName("love");
//遍历复选框的选择
for (i=0;i<selectObj.length;i++) {
    if (selectObj[i].checked) {
        j=1;
        break;
    }
}
if (j!=1) {
    alert("请选择至少一种爱好!");
    return false;
}
alert(selectObj[i].value);//显示所选择的第一项的内容
}
</script>
</head>
<body>
<form name="form1">
<input type="checkbox" name="love" value="旅游" >旅游
<input type="checkbox" name="love" value="体育" >体育<br/>
<input type="checkbox" name="love" value="购物" >购物
<input type="checkbox" name="love" value="影视" >影视
<input type="checkbox" name="love" value="音乐" >音乐
<input type="button" value="我的选择" onclick="mySelect()">
</form>
</html>
```

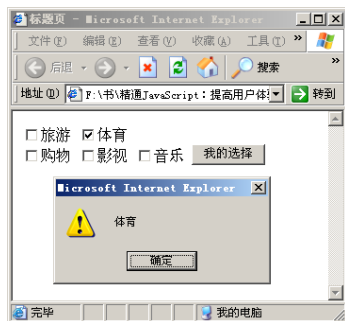


图 10-17 选择复选框后的效果

【运行效果】

选择复选框后的效果如图 10-17 所示。

【难点剖析】

本例的重点是获取用户选择的复选框，并显示复选框的值。复选框的值用“value”表示，在当前页面中必须唯一。“checked”属性用来判断复选框是否被选中。

10.17 改变 select 选中项的颜色特效

【实例描述】

默认的 select 选中项的颜色是黑底白字。有时为了实现与页面布局的统一，需要改变 select 选中项的颜色，本例介绍如何实现这个特效。

【实现代码】

```
<html>
<head>
```

```

<title>改变选中项的颜色</title>
</head>
<body >
<script language="javascript">
    function setColor(_parent, _child) {
        for (var i=0; i<_parent.options.length; i++) {           //遍历所有选项
            if (_parent.options[i] == _child) {
                _parent.options[i].style.color = 'yellow';       //颜色
                _parent.options[i].style.backgroundColor = 'blue'; //背景色
            } else {
                _parent.options[i].style.color = '';              //取消颜色
                _parent.options[i].style.backgroundColor = '';    //取消背景色
            }
        }
        document.body.focus();                                   //窗体获得焦点
    }
</script>
<select onchange="setColor(this, options[selectedIndex]);">
    <option style="color:yellow; background-color:blue;">选项 1</option>
    <option>选项 2</option>
    <option>选项 3</option>
    <option>选项 4</option>
    <option>选项 5</option>
</select>
</body>
</html>

```

【运行效果】

本例的运行效果如图 10-18 所示。



图 10-18 本例的运行效果

【难点剖析】

“options[索引]”用来获取 select 中的某项，“selectedIndex”表示选中项索引。“style.color”属性用来表示选择项的颜色，“style.backgroundColor”属性用来表示选择项的背景色。

第 11 章

颜色处理和菜单特效

本章导读

菜单是常用的网络导航工具，如果想让用户迅速了解网站的主要内容，网站一般会提供简洁明了的导航菜单。本章提供一些常用菜单样式的制作方法，并学习如何根据页面布局设计个性化的菜单。

11.1 背景颜色测试

【实例描述】

页面的颜色搭配一直是个难题。要想让页面的颜色视觉效果更好，可以使用本例提供的颜色测试代码。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
  <SCRIPT language=javascript>
    function makeArray(q)
    {
      for(i=1;i<=q;i++){this[i]=0}           //初始化数组
    }
    Colors=new makeArray(7);
    Colors[1]="00";Colors[2]="33";Colors[3]="66";
    Colors[4]="99";Colors[5]="CC";Colors[6]="FF";
  </SCRIPT>
</head>
<body>
  <center>
<table cellspacing="0" cellpadding="0">
<SCRIPT language=javascript>
//使用循环实现颜色的分段显示
for(i=1;i<=6;i++){
  for(j=1;j<=6;j++){
    for(k=1;k<=6;k++){
      var thiscolor=Colors[i]+Colors[j]+Colors[k];
      document.writeln("<tr><td height=20 bgcolor =\"#\" + thiscolor + \"\" align =
right><a href = \"");
      document.writeln("&quot;'\" onmouseover = \"document.bgColor='\"+thiscolor+ \"'\">\"
+thiscolor+\"</a></td></tr>");}}}
</SCRIPT>
</table></center>
</body>
</html>
```

【运行效果】

背景颜色的测试效果如图 11-1 所示。

【难点剖析】

本例的重点在于如何通过循环显示所有的颜色。用户将鼠标移动到颜色标码处，页面的景色就会发生变化。本例的技巧是使用数组保存了颜色的 16 进制代码，这样可以在页面中显示

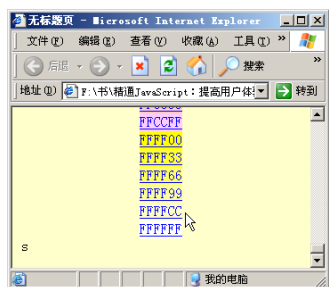


图 11-1 背景颜色的测试效果

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <SCRIPT LANGUAGE="JavaScript">
    function showRGB(f)
    {
      //获取用户输入的 RGB 值
      red = f.red.value;
      green = f.green.value;
      blue = f.blue.value;
      //将 RGB 转换为 16 进制 Hex 值
      hexcode = "#" + toHex(red) + toHex(green) + toHex(blue);
      document.bgColor = f.hexval.value = hexcode;
    }
    function toHex(d){
      if (isNaN(d)){
        d=0;
      }
      //16 进制转换方法
      var n=new Number(d).toString(16);
      return (n.length==1?"0"+n:n);
    }
  </script>
</head>
<body>
  <form name="rgbform">
    <b>请输入 RGB 颜色值(0 to 255)</b><br>
    Red:  <input type="text" name="red" size="5"><br>
    Green: <input type="text" name="green" size="5"><br>
    Blue: <input type="text" name="blue" size="5"><br>
    <input type="button" value="显示 Hex #" onClick="showRGB(this.form)">
    Hex 值为: <input type="text" name="hexval" size="7">
  </form>
</body>
</html>
```

颜色对应的 16 进制值。

11.2 RGB 颜色在线转换

【实例描述】

RGB 是一个用 10 进制数值表示颜色的方法，由 R、G、B 3 部分组成，每部分的值都不能超过 255。本例学习如何将颜色的 RGB 表示转换为 16 进制表示。

【运行效果】

RGB 颜色值转换效果如图 11-2 所示。

【难点剖析】

本例的难点在于进制间的转换。代码中使用“toHex”方法实现 10 进制到 16 进制的转换，但主要靠“toString”方法实现，此方法带一个参数表示要转换的进制。



图 11-2 RGB 颜色值转换效果

11.3 颜色切换板

【实例描述】

颜色切换板实现类似 Tab 控件的效果。用户通过选择某个颜色更改页面中指定区域的颜色。

【实现代码】

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>无标题文档</title>
</head>
<SCRIPT language=JavaScript>
function tdColor(num)
{
    var totalnum = 5;
    var tdStr;
    var i=0;
    for(i=1;i<=totalnum;i++){
        if(i==num){
            tdStr = "w"+i+".style.display='block'"; //显示指定的单元格
        }
        else{
            tdStr = "w"+i+".style.display='none'"; //隐藏指定的单元格
        }
        eval(tdStr);
    }
}
</SCRIPT>
<body>
<table width="405" height="279" border="0" cellpadding="0" cellspacing="0">
<tr>
    <td height="30" align="center" valign="middle" bgcolor="#CC3388"><a href="#"
onClick="tdColor(1);" >■</a></td>
    <td align="center" valign="middle" bgcolor="#00BBCC"><a href="#" onClick =
"tdColor(2);" >■</a></td>
    <td align="center" valign="middle" bgcolor="#BB5588"><a href="#" onClick =
"tdColor(3);" >■</a></td>
```

```
<td align="center" valign="middle" bgcolor="#0088BB"><a href="#" onClick =  
"tdColor(4);" >■</a></td>  
    <td align="center" valign="middle" bgcolor="#FFBB33"><a href="#" onClick=  
"tdColor(5);" >■</a></td>  
</tr>  
<tr id="TD1">  
    <td height="30" colspan="5" bgcolor="#CC3388">&nbsp;</td>  
</tr>  
<tr id="TD2" style="display:none;">  
    <td height="30" colspan="5" bgcolor="#00BBCC">&nbsp;</td>  
</tr>  
<tr id="TD3" style="display:none;">  
    <td height="30" colspan="5" bgcolor="#BB5588">&nbsp;</td>  
</tr>  
<tr id="TD4" style="display:none;">  
    <td height="30" colspan="5" bgcolor="#0088BB">&nbsp;</td>  
</tr>  
<tr id="TD5" style="display:none;">  
    <td height="30" colspan="5" bgcolor="#FFBB33">&nbsp;</td>  
</tr>  
</table>  
</body>  
</html>
```

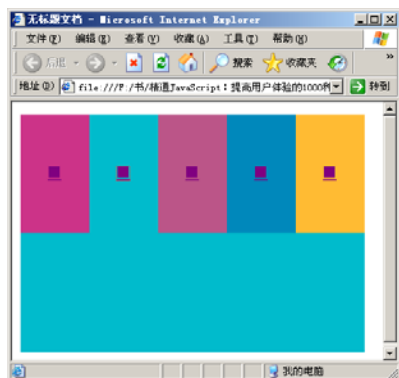


图 11-3 本例的运行效果

【运行效果】

本例的运行效果如图 11-3 所示。

【难点剖析】

本例的重点是页面的布局。实现类似 Tab 效果的其实是表格的单元格。界面的效果也是一个单元格合并。通过隐藏或显示这些合并单元格，可以实现 Tab 切换的效果。

11.4 下拉菜单

【实例描述】

分类下拉菜单可以尽可能地显示网站的导航信息。本例学习如何制作下拉式的导航菜单。

【实现代码】

```
<script language="javascript">  
//为各个菜单项定义鼠标事件  
startList = function()  
{  
    if (document.all&&document.getElementById) {  
        navRoot = document.getElementById("mynav");           //找到菜单根目录  
        for (i=0; i<navRoot.childNodes.length; i++)  
        {
```

```

        node = navRoot.childNodes[i];
        if (node.nodeName=="LI") {
            node.onmouseover=function() {
                this.className+=" over"; }
            node.onmouseout=function() {
                this.className=this.className.replace(" over", ""); }
        }
    }
}
}
window.onload=startList;
</script>

```

需要在 body 中添加一些菜单项，注意菜单项根目录的 ID 必须为“mynav”，代码如下所示：

```

<ul id="mynav">
<li><a href="#">首页</a></li>
<li><a href="#">帮助</a>
<ul>
<li><a href="#">历史</a></li>
<li><a href="#">团队</a></li>
<li><a href="#">办公室</a></li>
</ul>
</li>
<li><a href="#">客服</a>
<ul>
<li><a href="http://www.google.com">网页设计</a></li>
<li><a href="#">网络销售</a></li>
<li><a href="#">站点服务</a></li>
<li><a href="#">区域服务</a></li>
</ul>
</li>
<li><a href="#">联系方式</a>
<ul>
<li><a href="#">国家</a></li>
<li><a href="#">城市</a></li>
<li><a href="#">地址</a></li>
<li><a href="#">电话</a></li>
</ul>
</li>
</ul>

```

本例使用了大量的样式，样式的具体设计代码可参考随书光盘。

【运行效果】

下拉菜单的运行效果如图 11-4 所示。

【难点剖析】

本例在设计时使用了“ul”和“li”元素完成菜单项的设计。使用 JavaScript 中的“childNodes”

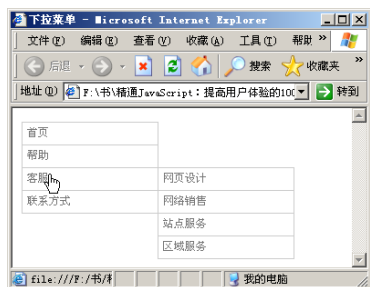


图 11-4 下拉菜单的运行效果

【实现代码】

```
<style type="text/css">
a.{ font: 9pt "宋体"; cursor: hand; font-size: 9pt ; color: #ffffff; text-decoration: none }
a:active{ font: 9pt "宋体"; cursor: hand; color: #FF0033 }
a.cc:hover{ font: 9pt "宋体"; cursor: hand; color: #FF0033}
.box{ font: 10pt "宋体"; position: absolute; background: #ccbbcc }
</style>
<script language="JavaScript">
function popUp()
{
newX = window.event.x + document.body.scrollLeft;
newY = window.event.y + document.body.scrollTop;
menu = document.all.menutable;
if ( menu.style.display == "" )
{
menu.style.display = "none" }
else {
menu.style.display = "";}
menu.style.pixelLeft = newX - 50;
menu.style.pixelTop = newY - 50;
}
</script>
```

需要在 body 的 onclick 事件中调用上面的方法，同时设置一个 table，实现菜单效果。代码如下所示：

```
<body onclick = popUp()>
<table id="menutable" class="box" style="display:none;width=120">
<tr>
<td>弹出菜单</td>
</tr>
<tr>
<td><a href="#" class="cc">本站搜索</a></td>
</tr>
<tr>
<td><a href="#" class="cc">本站帮助</a></td>
```

属性来获取每一个菜单项。注意“childNodes”是 JavaScript DOM 的重要组成部分，在 Ajax 应用中起了关键作用。

11.5 左键弹出式菜单

【实例描述】

在默认的网页中，使用右键可以弹出常用的操作菜单。本例学习使用左键弹出网站的导航菜单。

```
</tr>
<tr>
<td><a href="#" class="cc">当前更新</a></td>
</tr>
<tr>
<td><a href="#" class="cc">联系版主</a></td>
</tr>
</table>
</body>
```

【运行效果】

弹出菜单的效果如图 11-5 所示。

【难点剖析】

本例利用了 CSS 中的隐藏特性。将一个设置好的表格先隐藏起来，当用户单击鼠标左键时，设置“display”属性就可以实现表格的显示。

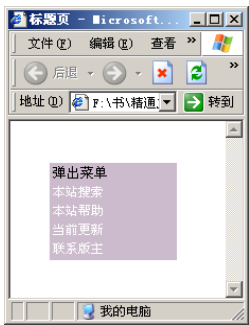


图 11-5 弹出菜单的效果

【实例描述】

目录样式的下拉菜单一般用于网站导航，就是以目录的形式列出本站的主要内容，当鼠标移动到菜单时，通过下拉形式展现更多的详细内容。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT language=javascript>
function out()
{
//判断鼠标是否移动到当前菜单或菜单中的项——第一个菜单的判断
if(window.event.toElement.id!="menuOne"    &&    window.event.toElement.id!= "link")
    menuOne.style.visibility="hidden";                //如果不是，则隐藏菜单
}
function out1()
{
//判断鼠标是否移动到当前菜单或菜单中的项——第二个菜单的判断
if(window.event.toElement.id!="menuTwo"    &&    window.event.toElement.id!="linkTwo")
    menuTwo.style.visibility="hidden";                //如果不是，则隐藏菜单
}
</SCRIPT>
</head>
<body>
<div id="back" onmouseout="out()"style="position:absolute;top:180;left:310; width:
160;height:40;z-index:1;visibility:visible;">
<span id="menubar" onmouseover="menuOne.style.visibility='visible'">
```



```
<font color=red size=2>本站导航</span>
<div border=1 id="menuOne" style="position:absolute;top:15;left:0;width:70; height:
10;z-index:2;visibility:hidden;">
  <a id="link" href="#">软件下载</a>
  <a id="A1" href="#">工具下载</a>
  <a id="A2" href="#">书籍介绍</a>
  <a id="A3" href="#">代码题库</a>
  <a id="A4" href="#">先进技术</a>
  <a id="A5" href="#">技术人物</a>
</div>
</div>
<div id="Div1" onmouseout="out1()" style="position:absolute;top:181px;left: 399px;
width:160;height:40;z-index:3;visibility:visible;">
  <span id="Span1" onmouseover="menuTwo.style.visibility='visible'">
  <font color=red size=2>服务导航</span>
  <div border=1 id="menuTwo" style="position:absolute;top:15;left:0;width:70; height:
10;z-index:4;visibility:hidden;">
    <a id="linkTwo" href="#">软件下载</a>
    <a id="A7" href="#">工具下载</a>
    <a id="A8" href="#">书籍介绍</a>
    <a id="A9" href="#">代码题库</a>
    <a id="A10" href="#">先进技术</a>
    <a id="A11" href="#">技术人物</a>
  </div>
</div>
</body>
</html>
```

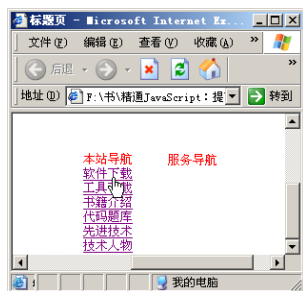


图 11-6 目录样式的下拉效果

【运行效果】

目录样式的下拉效果如图 11-6 所示。

【难点剖析】

本例的重点是鼠标的两个事件“onmouseover”和“onmouseout”。当鼠标移动到菜单时，首先判断菜单的 id，如果鼠标的确是指向菜单，则设置菜单为可见；如果不是，则设置菜单为不可见，这里用了菜单的“visibility”属性。

11.7 网页中的选项卡

【实例描述】

选项卡是 C/S（Client/Server）应用程序中常见的控件，但网页中并无此控件。本例学习使用 JavaScript 和 CSS 实现 B/S（Browser/Server）应用程序的选项卡。

【实现代码】

```
<script language="JavaScript">
```



```

function public_Labels(label1, label2, label3, label4, label5, label6, label7){
t1.innerText = label1;
t2.innerText = label2;
t3.innerText = label3;
t4.innerText = label4;
t5.innerText = label5;
t6.innerText = label6;
t7.innerText = label7;
}
//切换选项卡时显示选项卡的内容
function public_Contents(contents1, contents2, contents3, contents4, contents5,
contents6, contents7){
t1Contents.innerHTML = contents1;
t2Contents.innerHTML = contents2;
t3Contents.innerHTML = contents3;
t4Contents.innerHTML = contents4;
t5Contents.innerHTML = contents5;
t6Contents.innerHTML = contents6;
t7Contents.innerHTML = contents7;
init();
}
//默认显示选项卡 1
function init(){
tabContents.innerHTML = t1Contents.innerHTML;
}
//更换选项卡时的方法
var currentTab;
var tabBase;
var firstFlag = true;
function changeTabs(){
if(firstFlag == true){
currentTab = t1;
tabBase = t1base;
firstFlag = false;
}
}
//当用户单击选项卡时, 修改样式及内容
if(window.event.srcElement.className == "tab")
{
currentTab.className = "tab";
tabBase.style.backgroundColor = "white";
currentTab = window.event.srcElement;
tabBaseID = currentTab.id + "base";
tabContentID = currentTab.id + "Contents";
tabBase = document.all(tabBaseID);
tabContent = document.all(tabContentID);
currentTab.className = "selTab";
tabBase.style.backgroundColor = "";
tabContents.innerHTML = tabContent.innerHTML;
}

```



```
}}  
</script>
```

本例中使用了大量的样式表，具体代码可参考随书光盘。

【运行效果】

选项卡效果如图 11-7 所示。

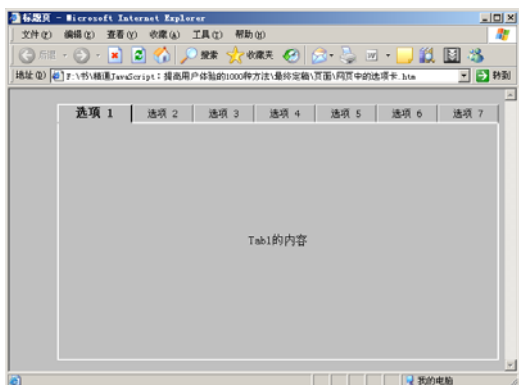


图 11-7 选项卡效果

【难点剖析】

本例的重点是选项卡的样式和切换选项卡的事件。其中使用了“window.event.srcElement”来获取选择的 Tab。

11.8 静态导航菜单

【实例描述】

静态导航菜单一般用于内容比较固定的网站，大多都是小型网站。菜单的内容都是预先设置好的，不允许在运行时动态改变。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
</head>  
<BODY bgcolor="#ffffff" OnLoad="setVariables();checkLocation()">  
  
<script language="JavaScript">  
  
function setVariables() {  
if (navigator.appName == "Netscape") { //浏览器是 Netscape 的情况  
v=".top=";  
dS="document.";  
sD=" " ;  

```

```

        y="window.pageYOffset";
    }
    else {
        // 浏览器是 IE 的情况
        v=".pixelTop=";
        dS="";
        sD=".style";
        y="document.body.scrollTop"; // 如果页面有滚动条, 获取滚动条的顶端位置
    }
}
function checkLocation() {
    object="object1"; // 获取指定的 div
    yy=eval(y);
    eval(dS+object+sD+v+yy); // 连接字符串, 旨在让静态导航菜单一直在页面滚动条的上端
    setTimeout("checkLocation()",10);
}
</script>
<div id="object1" style="position:absolute; visibility:show; left:0px; top:0px;
z-index:5">
    <table width=150 border=0 cellpadding=20 cellspacing=0 >
        <tr>
            <td bgcolor="#EAEAEA">
                <center>
                    静态导航菜单
                </center>
            </td>
        </tr>
        <tr>
            <td bgcolor="#EAEAEA"><a href="http://www.microsoft.com" >微软网站</a></td>
        </tr>
        <tr>
            <td bgcolor="#EAEAEA"><a href="http://www.yahoo.com" >雅虎网站</a></td>
        </tr>
        <tr>
            <td bgcolor="#EAEAEA"><a href="http://www.baidu.com">百度搜索</a></td>
        </tr>
        <tr>
            <td bgcolor="#EAEAEA"><a href="http://mail.263.com" >263 邮局</a></td>
        </tr>
    </table>
</div>
</body>
</html>

```

【运行效果】

静态菜单的效果如图 11-8 所示。

【难点剖析】

本例中页面的布局很重要, 使用一个 table 封装所有的静态链接, 然后将 table 放在 div 层

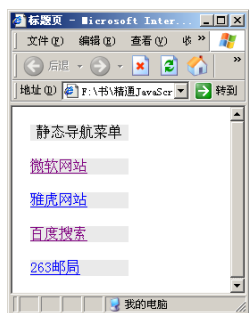


图 11-8 静态菜单的效果

中。使用“checkLocation”方法设置层的位置一直显示在滚动条的顶端位置。

11.9 烟花效果的下拉菜单

【实例描述】

在实际网站建设中，经常需要使用 CSS+JavaScript 的形式实现一些特殊的效果，本例中的“烟花”菜单就是利用 CSS 的一些特性。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <style>
    #iewrap{
      position:relative;
      height:30px
    }
    #iewrap2{
      position:absolute;
    }
    #dropmenu03{
      filter:revealTrans(Duration=1.5,Transition=12);
      visibility:hide;
    }
    a:hover { color: #FF0000}
    body { font-family: "宋体"; font-size: 9pt; text-decoration: none}
    a { font-family: "宋体"; font-size: 9pt; text-decoration: none}
  </style>
</head>
<body>
  <ilayer id="dropmenu01" height=35px>
    <layer id="dropmenu02" visibility=show>
      <span id="iewrap">
        <span id="iewrap2" onClick="dropit2();event.cancelBubble=true;return false">
          <font face="宋体"><a href="#">单击此处出现菜单</a></font>
        </span>
      </span>
    </layer>
  </ilayer>
  <script language="JavaScript">
    //关闭 "fade"（消隐）效果，将如下参数设置成 false
    var enableeffect=true;
    //菜单项
    var selection=new Array();
```

```

selection[0]='<font face="宋体"><a href="#">第一个菜单</a></font><br>';
selection[1]='<font face="宋体"><a href="#">第二个菜单</a></font><br>';
selection[2]='<font face="宋体"><a href="#">第三个菜单</a></font><br>';
selection[3]='<font face="宋体"><a href="#">第四个菜单</a></font><br>';
selection[4]='<font face="宋体"><a href="#">第五个菜单</a></font>';

if (document.layers)
document.dropmenu01.document.dropmenu02.visibility='show';
function dropit2()
{
if (document.all){
dropmenu03.style.left=document.body.scrollLeft+event.clientX-event.offsetX;
dropmenu03.style.top=document.body.scrollTop+event.clientY-event.offsetY+18;
if (dropmenu03.style.visibility=="hidden"){
if (enableeffect)
dropmenu03.filters.revealTrans.apply();           //配置特殊效果
dropmenu03.style.visibility="visible";
if (enableeffect)
dropmenu03.filters.revealTrans.play();           //演示特殊效果
}
else{
hidemenu();           //隐藏菜单
}}}
function dropit(e)
{
if (document.dropmenu03.visibility=="hide")
document.dropmenu03.visibility="show";
else
document.dropmenu03.visibility="hide";
document.dropmenu03.left=e.pageX-e.layerX;
document.dropmenu03.top=e.pageY-e.layerY+19;
return false;
}
//隐藏菜单的方法
function hidemenu()
{
if (enableeffect)
dropmenu03.filters.revealTrans.stop();           //停止特效
dropmenu03.style.visibility="hidden";
}
function hidemenu2()
{document.dropmenu03.visibility="hide";}
if (document.layers){
document.dropmenu01.document.dropmenu02.captureEvents(Event.CLICK);
document.dropmenu01.document.dropmenu02.onclick=dropit;}
else if (document.all)
document.body.onclick=hidemenu;
</script>
<div id="dropmenu03" style="position:absolute;left:0;top:0;layer- background-color:

```



```
seashell;background-color:seashell;width:100;visibility:hidden;border:1px solid black;
padding:0px">
<script language="JavaScript1.2">
if (document.all)
dropmenu03.style.padding='4px'
for (i=0;i<selection.length;i++)
document.write(selection[i])
</script>
</div>
<script language="JavaScript">
if (document.layers){
document.dropmenu03.captureEvents(Event.CLICK);
document.dropmenu03.onclick=hidemenu2;
}
</script>
</body>
</html>
```



图 11-9 下拉菜单的运行效果

【运行效果】

下拉菜单的运行效果如图 11-9 所示。

【难点剖析】

本例的重点在于“revealTrans”属性，其提供了多种滤镜效果的 CSS 特效。其使用语法如下所示：

```
Filter:revealtrans(duration=转换的秒数, transition=转换的类型)
```

“revealTrans”还提供三种方法：apply、play 和 stop，分别代表应用特效、演示特效和停止特效。

11.10 网络导航条

【实例描述】

网络导航条和网络导航菜单一样，用来让用户了解网站的所有功能。本例提供一个效果非常酷的网络导航条。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<style>
/* 先把这个 myMenu 的样式放到 css 里 */
.myMenu td{font-size:12px;font-family:verdana,arial;font-weight:bolder; color:
#ffffff;border:1px solid #336699;background:#336699;filter:blendtrans (duration=0.5);
```

```

cursor:hand;text-align:center;}
</style>
<script>

//把事件动作绑定到菜单上
function addMenu(objid){
var tds=objid.getElementsByTagName('td');
for(var i=0;i<tds.length;i++){
    with(tds[i]){
        onmouseover=function(){
            with(this){
                filters[0].apply();
                style.background='#FEBD20';           //鼠标移上时的背景颜色
                style.border='1px solid #ffffff';       //边框
                style.color='black';                   //文字颜色
                filters[0].play();
            }
        }
        onmouseout=function(){
            with(this){
                filters[0].apply();
                style.background='#336699';           //鼠标离开时的背景颜色
                style.border='1px solid #336699';       //边框
                style.color='#ffffff';                 //文字颜色
                filters[0].play();
            }
        }
    }
}
}
</script>
<!--菜单从这里开始，注意要把 class 设置成和 css 里相同的，还要设一个 id-->
<table class="myMenu" id="menuTbl" width="500" cellpadding="1" cellspacing=" 4"
border="0" bgcolor="#00000" align="center">
<tr>
<td>网站介绍</td>
<td>开发文档</td>
<td>下载软件</td>
<td>开源管理</td>
<td>网站服务</td>
</tr>
</table>
<script>addMenu(menuTbl); //在上面这个 table 结束的地方执行事件动作的绑定，这里的这个//menuTbl
就是那个 table 的 id
</script></body>
</html>

```

【运行效果】

本例的运行效果如图 11-10 所示。

【难点剖析】

本例的重点是如何动态地为 td 标签添加事件。网络导航条由一个普通的表格组成，初始状态下没有设置任何事件，只有一个简单的样式。使用“addMenu”方法，为表格中所有的 td 动态绑定了“onmouseover”和“onmouseout”事件，通过设置边框、背景和文字颜色改变单元格的效果。



图 11-10 本例的运行效果

11.11 隐藏式菜单

【实例描述】

菜单是网站的常用导航形式，为了保证页面的整体效果，有时候需要将菜单隐藏起来，当用户需要的时候再显示出来。本例学习如何制作隐藏式菜单。

【实现代码】

```
<SCRIPT language="javascript">
//状态栏显示信息
function MM_displayStatusMsg(msgStr)
{
    status=msgStr;
    document.MM_returnValue = true;
}
//高亮显示
function highlight(x)
{
    document.forms[x].elements[0].focus()
    document.forms[x].elements[0].select()
}
//菜单弹出时的处理
function MM_jumpMenu(targ,selObj,restore)
{
    eval(targ+".location='"+selObj.options[selObj.selectedIndex].value+"'");
    if (restore) selObj.selectedIndex=0;
}
var NS;
IE=document.all;
NS=document.layers;
```



```

hdrFontFamily="Verdana";
hdrFontSize="2";
hdrFontColor="white";
hdrBGColor="#CCCCCC";
linkFontFamily="Verdana";
linkFontSize="2";
linkBGColor="white";
linkOverBGColor="#CCCCCC";
linkTarget="_top";
YOffset=60;
staticYOffset=20;
menuBGColor="#CCCCCC";
menuIsStatic="no";
menuHeader="主流索引"
menuWidth=150;
staticMode="advanced"
barBGColor="#C0C0C0";
barFontFamily="Verdana";
barFontSize="2";
barFontColor="white";
barText="导航菜单";
//鼠标移走时的菜单处理
function moveOut()
{
if (window.cancel)
{ cancel="";}
if (window.moving2)
{ clearTimeout(moving2);
moving2="";
}
if ((IE && ssm2.style.pixelLeft<0)|| (NS && document.ssm2.left<0))
{
if (IE) {ssm2.style.pixelLeft += (5%menuWidth);
}
if (NS)
{
document.ssm2.left += (5%menuWidth);
}
moving1 = setTimeout('moveOut()', 5)
}
else {
clearTimeout(moving1)
}
};
function moveBack() {
cancel = moveBack1()
}
function moveBack1() {

```



```
if (window.moving1) {
    clearTimeout(moving1)
}
if ((IE && ssm2.style.pixelLeft > (-menuWidth)) || (NS && document.ssm2.left > (-150))) {
    if (IE) {ssm2.style.pixelLeft -= (5%menuWidth);
}
if (NS) {
    document.ssm2.left -= (5%menuWidth);
}
moving2 = setTimeout('moveBack1()', 5)}
else {
    clearTimeout(moving2)
}
};
lastY = 0;
//根据浏览器类型设置菜单模式
function makeStatic(mode) {
    if (IE) {winY = document.body.scrollTop;var NM=ssm2.style
}
if (NS) {winY = window.pageYOffset;var NM=document.ssm2
}
if (mode=="smooth") {
    if ((IE|NS) && winY!=lastY) {
        smooth = .2 * (winY - lastY);
        if(smooth > 0) smooth = Math.ceil(smooth);
        else smooth = Math.floor(smooth);
        if (IE) NM.pixelTop+=smooth;
        if (NS) NM.top+=smooth;
        lastY = lastY+smooth;
    }
    setTimeout('makeStatic("smooth")', 1)
}
else if (mode=="advanced") {
    if ((IE|NS) && winY>YOffset-staticYOffset) {
        if (IE) {NM.pixelTop=winY+staticYOffset
        }
        if (NS) {NM.top=winY+staticYOffset
        }
    }
    else {
        if (IE) {NM.pixelTop=YOffset}
        if (NS) {NM.top=YOffset-7 }
    }
    setTimeout('makeStatic("advanced")', 1)
}
}
//根据浏览器类型初始化菜单
function init() {
```

```

    if (IE) {
        ssm2.style.pixelLeft = -menuWidth;
        ssm2.style.visibility = "visible"
    }
    else if (NS) {
        document.ssm2.left = -menuWidth;
        document.ssm2.visibility = "show"
    }
    else {
        alert('浏览器类型判断错误!')
    }
}
//当用户选择菜单时, 在状态栏提示导航地址
function MM_displayStatusMsg(msgStr) {
    status=msgStr;
    document.MM_returnValue = true;

}
//判断浏览器类型
if (IE) {document.write('<DIV ID="ssm2" style="visibility:hidden;Position :
Absolute ;Left : 0px ;Top : '+YOffset+'px ;Z-Index : 20;width:1px" onmouseover ="moveOut()"
onmouseout="moveBack()">')}
    if (NS) {document.write('<LAYER visibility="hide" top="'+YOffset+'" name= "ssm2"
bgcolor="'+menuBGColor+'" left="0" onmouseover="moveOut()" onmouseout= "moveBack()">')}
    tempBar="";
    for (i=0;i<barText.length;i++) {
        tempBar+=barText.substring(i, i+1)+"<BR>"
    }
    document.write('<table border="0" cellpadding="0" cellspacing="1" width= ''+
(menuWidth+16+2)+'" bgcolor="'+menuBGColor+'"><tr><td bgcolor="'+hdrBGColor+'" WIDTH=
'+menuWidth+'"> <font face="'+hdrFontFamily+'" Size="'+hdrFontSize+'" COLOR="
'+hdrFontColor+'"><b>'+menuHeader+'</b></font> </td><td align="center" rowspan="100"
width="16" bgcolor="'+barBGColor+'"><p align="center"><font face="'+barFontFamily+'"
Size="'+barFontSize+'" COLOR= "'+barFontColor+'"><B>'+tempBar+'</B></font></p></TD> </tr>')
    //添加菜单内容项的方法
    function addItem(text, link, target)
    {
        if (!target) {target=linkTarget}
        document.write('<TR><TD BGCOLOR="'+linkBGColor+'" onmouseover="bgColor= \''+
linkOverBGColor+'\' onmouseout="bgColor=\''+linkBGColor+'\'"><ILAYER> <LAYER onmouseover
="bgColor=\''+linkOverBGColor+'\' onmouseout="bgColor= \''+linkBGColor+'\'" WIDTH=
"100%"><FONT face="'+linkFontFamily+'" Size="'+ linkFontSize+'"> <A HREF="'+link+'
target="'+target+' " CLASS="ssm2Items"> '+text+'</A></FONT></LAYER></ILAYER></TD></TR>')
    }

    //为菜单添加分类标题的方法
    function addHdr(text)
    {
        document.write('<tr><td bgcolor="'+hdrBGColor+'" WIDTH="140"> <font face="'+

```

```
hdrFontFamily+'" Size="'+hdrFontSize+'" COLOR="'+hdrFontColor+'">
<b>'+text+'</b></font></td></tr>')
    }
    //下面是菜单内容和指定的导航地址
    addItem('初级搜索', 'http://www.google.com', '');
    addItem('中级搜索', 'http://www.google.com', '');
    addItem('搜索原理', 'http://www.google.com', '');
    addItem('深入搜索', 'http://www.google.com', '');
    addItem('垂直搜索', 'http://www.google.com', '_blank');
    addHdr('帮助'); //菜单标题
    addItem('会员专属论坛', 'http://www.google.com', '_blank');
    document.write('<tr><td bgcolor="'+hdrBGColor+'"><font size="0" face="Arial"> </font>
</td></tr></table>')
    if (IE) {document.write('</DIV>')}
    if (NS) {document.write('</LAYER>')}
    if ((IE|NS) && (menuIsStatic=="yes"&&staticMode)) {makeStatic(staticMode);}
    window.onload=init;
</SCRIPT>
```

本例中用到的样式代码如下所示：

```
<STYLE>
#ssm2 A
{ FONT-SIZE: 12px; COLOR: #808080; FONT-FAMILY: verdana; TEXT-DECORATION: none}
#ssm2 A:hover
{ COLOR: #ccff33}
</STYLE>
```



图 11-11 隐藏式菜单的显示效果

【运行效果】

隐藏式菜单的显示效果如图 11-11 所示。

【难点剖析】

本例中的主要方法是“addItem”和“addHdr”。“addItem”方法主要完成菜单项的添加，“addHdr”方法则为菜单添加分类标题，如本例中的“主流索引”和“帮助”。本例中还需要注意的是如何判断 IE 浏览器和非 IE 浏览器，因为针对不同浏览器的 JavaScript 对象有所区别。

11.12 仿 Flash 菜单

【实例描述】

用 JavaScript 可以很轻松地构建菜单。本例学习制作一个具有 Flash 效果的菜单。

【实现代码】

```
<script language=javascript>
```

```

// 绑定菜单的事件
function attachXMenu(objid)
{
    var tds=objid.getElementsByTagName('td');
    for(var i=0;i<tds.length;i++){
        with(tds[i]){
            onmouseover=function(){
                with(this){
                    filters[0].apply();
                    style.background='#66CCFF';           //鼠标移上时的背景颜色
                    style.border='1px solid #ffffff';      //边框
                    style.color='black';                   //文字颜色
                    filters[0].play();
                }
            }
            onmouseout=function(){
                with(this){
                    filters[0].apply();
                    style.background='#336699';           //这是鼠标离开时的背景颜色
                    style.border='1px solid #336699';      //边框
                    style.color='#ffffff';                 //文字颜色
                    filters[0].play();
                }
            }
        }
    }
}
</script>

```

需要添加菜单的样式表，代码如下所示：

```

<style>
.xmenu td{font-size:12px;font-family:verdana,arial;font-weight:bolder; color: #ffffff;
border:1px solid #336699;background:#336699;filter:blendtrans(duration=0.5);
cursor:hand;text-align:center;}
</style>

```

需要在 body 中添加一个表格，用来制作菜单，代码如下所示：

```

<table class="xmenu" id="xmenu1" width="100" cellpadding="1" cellspacing="4" border="0"
bgcolor="#336699" align="center">
    <tr><td>Name</td></tr>
    <tr><td>Age</td></tr>
    <tr><td>Address</td></tr>
    <tr><td>City</td></tr>
    <tr><td>Grade</td></tr>
    <tr><td>Teacher</td></tr>
</table>
<script>attachXMenu(xmenu1);</script>

```



图 11-12 仿 flash 菜单的效果

【运行效果】

仿 flash 菜单的效果如图 11-12 所示。

【难点剖析】

本例中有样式变换和动态事件两个重点。为了突出 flash 效果，当鼠标移动到菜单上时，菜单的样式需要发生变化，这归功于样式表和“attachXMenu”方法。在“attachXMenu”中，使用“onmouseover=function()”语句为菜单添加了动态事件。

11.13 滚动导航菜单

【实例描述】

导航菜单可以是隐藏式，可以是静止式，还可以是下拉式。本例学习一种滚动式的导航菜单。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language=javascript>
var index = 9
link = new Array(8);
text = new Array(8);
link[0] = 'link1.htm' //保存链接的数组
link[1] = 'link2.htm'
link[2] = 'link3.htm'
link[3] = 'link1.htm'
link[4] = 'link2.htm'
link[5] = 'link3.htm'
link[6] = 'link1.htm'
link[7] = 'link2.htm'
link[8] = 'link3.htm'
text[0] = '链接说明一' //保存说明的数组
text[1] = '链接说明一'
text[2] = '链接说明一'
text[3] = '链接说明一'
text[4] = '链接说明一'
text[5] = '链接说明一'
text[6] = '链接说明一'
text[7] = '链接说明一'
text[8] = '链接说明一'
//实现滚动的脚本
document.write("<marquee scrollamount='1' scrolldelay='100' direction='up' width='150' height='150'>");
for (i=0;i<index;i++){
```

```
document.write ("&nbsp;<a href="+link[i]+" target='_blank'>");//循环输出链接内容
document.write (text[i] + "</A><br>");
}
document.write ("</marquee>")
</script>
</body>
</html>
```

【运行效果】

滚动导航菜单的效果如图 11-13 所示。

【难点剖析】

本例的重点是将“marquee”标签和“a”标签结合。首先将所有的链接 URL 和链接说明封装在两个数组中，然后在“marquee”标签中使用循环，循环“a”标签，并指定每个“a”标签的 URL 和链接说明。

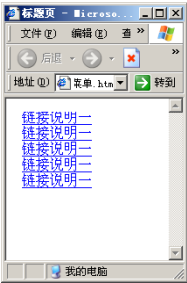


图 11-13 滚动导航菜单的效果

11.14 幻灯片式的导航菜单

【实例描述】

导航菜单用一个 div 封装，每次只显示一个菜单，并实现菜单变化时的幻灯片效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style>
#linkView{
position:relative;
layer-background-color:black;
width:400;
height:12;
; font-family: "宋体"; font-size: 9pt}
#subtickertape{
background-color:black;
position:absolute;
border: 1px solid black;
width:400;
height:12;
; font-family: "宋体"; font-size: 9pt
}
.subtickertapefont{
font:bold 9pt "宋体";
text-decoration:none;
color:white;
}
.subtickertapefont a{
```



```

color:white;
text-decoration:none;
; font-family: "宋体"; font-size: 9pt}
</style>
</head>
<body bgcolor="#fef4d9" onload="if (document.all||document.layers) {regenerate2();
update()}">
<div id="linkView">
<div id="subtickertape" class="subtickertapefont">初始化...</div>
</div>
<script language="JavaScript">
//默认速度 3 秒，可以修改速度快慢
var speed=3000
var news=new Array()
news[0]="<a href='index1.htm'>我的连接 1</a>" //创建链接树数组
news[1]="<a href='index2.htm'>我的连接 2</a>"
news[2]="<a href='index3.htm'>我的连接 3</a>"
news[3]="<a href='index4.htm'>我的连接 4</a>"
news[4]="<a href='index5.htm'>我的连接 5</a>"
news[5]="<a href='index6.htm'>我的连接 6</a>"
news[6]="<a href='index7.htm'>我的连接 7</a>"

//显示的信息内容按照格式添加
i=0
if (document.all)
tickerobject=document.all.subtickertape.style
else
tickerobject=document.linkView.document
function regenerate(){
window.location.reload() //重新加载页面
}
function regenerate2(){
if (document.layers) //非 IE 浏览器时的定时器
setTimeout("window.onresize=regenerate",450)
}

function update(){
BgFade(0xff,0xff,0xff, 0x00,0x00,0x00,10);
if (document.layers){
document.linkView.document.subtickertape.document.write('<span class=
"subtickertapefont">'+news[i]+'</span>')
document.linkView.document.subtickertape.document.close()
}
else
document.all.subtickertape.innerHTML=news[i]//动态显示链接（使用数组索引找到要显示的链接）
if (i<news.length-1) //判断是否已经显示完所有链接
i++ //如果不是，则显示下个链接
else
i=0 //否则从头开始

```



```

setTimeout("update()",speed)
}

function BgFade(red1, grn1, blu1, red2,
grn2, blu2, steps) {
sred = red1; sgrn = grn1; sblu = blu1;      //颜色的设置
ered = red2; egrn = grn2; eblu = blu2;
inc = steps;
step = 0;
RunFader();
}

function RunFader() {
var epct = step/inc;
var spct = 1 - epct;
if (document.layers)
tickerobject.bgColor =
Math.floor(sred * spct + ered *
epct)*256*256 +
Math.floor(sgrn * spct + egrn * epct)*256 +
Math.floor(sblu * spct + eblu * epct);
else                                     //背景色的渐变效果，注意颜色的设置
tickerobject.backgroundColor=
Math.floor(sred * spct + ered *
epct)*256*256 +
Math.floor(sgrn * spct + egrn * epct)*256 +
Math.floor(sblu * spct + eblu * epct);
if ( step < inc ) {
setTimeout( 'RunFader()',50);          //循环执行渐变效果
}
step++;
}
</script>
</body>
</html>

```

【运行效果】

幻灯片式的导航菜单效果如图 11-14 所示。

【难点剖析】

本例的重点是链接的显示和特效的实现。所有链接都保存在“news”数组中。通过循环使用数组索引逐个显示链接的内容，然后使用“RunFader”方法，实现链接更改时的幻灯片效果。

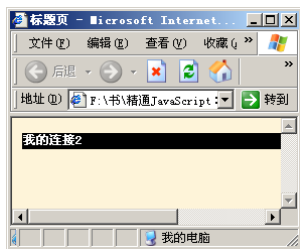


图 11-14 幻灯片式的导航菜单效果

11.15 类似 QQ 主界面的菜单

【实例描述】

QQ 主界面其实是 C/S 实现的菜单。本例通过 JavaScript 实现一个弹出式的菜单，效果与



QQ 主界面类似。

【实现代码】

```
<script language="JavaScript">
var titleHeight = 22;                //标题的高度
var bodyHeight = 160;                //菜单内容项的高度
var groupcount = 4;                  //组的个数
var step = 3;                        //移动速度
//显示菜单的内容
function showDiv(obj1, obj2)
{
    //以下循环为改变标题的背景颜色
    for(i=0;i<document.all.tags("td").length;i++)
    {
        if (document.all.tags("td")[i].className == 'headtd')
            document.all.tags("td")[i].bgColor = '#cccc66';
    }
    obj2.bgColor = '#cccc66';
    moveme(obj1);
}
//移动菜单内容
function moveme(obj)
{
    idnumber = parseInt(obj.id.substr(4));
    objtop = titleHeight * (idnumber - 1);
    objbottom = bodyHeight + titleHeight * (idnumber - 2);
    currenttop = parseInt(obj.style.top);
    if (currenttop >= objbottom)
    {
        //检验出每一个应该向上移动的层
        countid = 1;
        for(i=0;i<document.all.tags("div").length;i++)
        {
            if (document.all.tags("div")[i].id == 'item'+countid+'body')
            {
                obj = document.all.tags("div")[i];
                objtop = titleHeight * (countid - 1);
                moveup(obj,objtop);
                if (countid == idnumber)
                    break;
                countid++;
            }
        }
    }
    else if ((currenttop <= objtop) && (idnumber < groupcount))
    {
        //检验出每一个应该向下移动的层
        idnumber++;
        countid = groupcount;
    }
}
```

```

for(i=document.all.tags("div").length-1;i>=0;i--)
{
    if (document.all.tags("div")[i].id == 'item'+countid+'body')
    {
        obj = document.all.tags("div")[i];
        objbutton = bodyHeight + titleHeight * (countid - 2);
        movedown(obj,objbutton);
        if (countid == idnumber)
            break;
        countid--;
    }
}
}
}
//向上移动的方法
function moveup(obj,objtop)
{
    currenttop = parseInt(obj.style.top);
    if (currenttop > objtop)
    {
        obj.style.top = currenttop - step;
        window.setTimeout('moveup('+obj.id+', '+objtop+')',1)
    }
}
//向下移动的方法
function movedown(obj,objbutton)
{
    currenttop = parseInt(obj.style.top);
    if (currenttop < objbutton)
    {
        obj.style.top = currenttop + step;
        window.setTimeout('movedown('+obj.id+', '+objbutton+')',1)
    }
}
</script>

```

【运行效果】

菜单的运行效果如图 11-15 所示。

【难点剖析】

本例的重点是菜单的布局。使用 div 作为菜单的标题，使用 table 作为菜单的内容，通过设置样式实现菜单的下拉和收缩效果。

11.16 三级联动菜单（一）

【实例描述】

三级联动菜单是常用的一种选择方式，通过这种方式，用户可以更方便地选择数据，同时



图 11-15 菜单的运行效果



还能降低选择的失误率。本例学习如何制作三级联动菜单，实现方法是普通事件的捕获。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<STYLE>
body { font-size: 11px; font-family: Verdana;background:#ecec;color:#666666;}
select { font-size: 11px; font-family: Verdana;vertical-align: middle;margin:
3px;background:#ecec;color:#666666;}
</STYLE>
<div id="myDiv">&nbsp;</div>
<SCRIPT LANGUAGE="JavaScript" DEFER>
var cMenu = ["上海","北京","深圳","济南"]
var cValue = ["shanghai","beijing","shenzhen","jinan"]
var aMenu = [["杨浦区","徐汇区","黄浦区","浦东新区"],["丰台区","海淀区"],["福田","宝安"],["市
中","历下"]]
var aValue = [["yp","xh","hp","pd"],["ft","hd"],["ft","ba"],["sz","lx"]]
var dMenu = [[["市光新村","工农三村"],["徐汇新村","徐汇高楼"],["黄浦楼字","外滩风景"],["浦东地
铁","浦东机场"]],["丰台体育馆","造甲村"],["亚运村"]],["莲花山","商报大厦"],["深圳宝安机场","宝安开
发区"]],["山庄宾馆","山庄大酒店"],["千佛山东门","千佛山医院"]]]

var oDiv = document.all.myDiv;
var ocMenu = document.createElement("<SELECT name='city'>"); //创建城市列表框
var oaMenu = document.createElement("<SELECT name='cityArea'>"); //创建区域列表框
var odMenu = document.createElement("<SELECT name='cityAddress'>"); //创建地址列表框
with(oDiv)appendChild(ocMenu),appendChild(oaMenu),appendChild(odMenu);
//将以上 3 个列表框添加到 div 中

createMainOptions();
createSubOptions(0);
createSub2Options(0,0);

ocMenu.onchange = function() { //绑定城市下拉框的选择事件
createSubOptions(this.selectedIndex);createSub2Options(this.selectedIndex,
oaMenu.selectedIndex);};
oaMenu.onchange = function() { //绑定区域下拉框的选择事件
createSub2Options(ocMenu.selectedIndex,this.selectedIndex);};

function createMainOptions() {
for(var i=0;i<cMenu.length;i++)ocMenu.options[i] = new Option(cMenu[i],cValue[i]);
//填充城市列表框内容
}
function createSubOptions(j) {
with(oaMenu) {
length=0; //填充区域列表框内容
for(var i=0;i<aMenu[j].length;i++)oaMenu.options[i] = new Option(aMenu[j][i],
aValue[j][i]);

```

```

    }
}
function createSub2Options(j,k) {
    with(odMenu) {
        length=0; //填充地址列表框内容
        for(var i=0;i<dMenu[j][k].length;i++)odMenu.options[i] = new Option(dMenu[j]
[k][i]);
    }
}
ocMenu[1].selected=true;
ocMenu.fireEvent("onchange");
</SCRIPT>
</head>
<body>
</body>
</html>

```

【运行效果】

三级联动菜单的效果如图 11-16 所示。

【难点剖析】

本例中使用数组定义了 3 个级别下拉框的内容, 然后使用“createElement”动态创建了 3 个下拉框, 并使用“appendChild”将下拉框添加到指定的 div 层中。为了实现联动效果, 还动态设置了这 3 个下拉框的“onchange”事件。

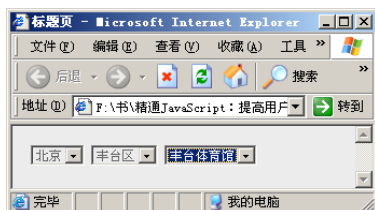


图 11-16 三级联动菜单的效果

11.17 三级联动菜单（二）

【实例描述】

三级联动菜单一般用在多项选择上, 如果用户不选择第一级菜单, 就无法实现第二级菜单的选择。本例学习制作三级联动菜单, 与前例不同的是, 本例的实现方法是通过封装方法的调用。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
</head>
<body>
<form method="post" name="form1">
<select name="select1" onchange="select()"></select>
<select name="select2"></select>
<select name="select3"></select>
</form>
<script language="javascript">
var arrText = new Array(3);

```



```
var arrValue = new Array(arrText.length);
//设置 3 个下拉框的值
function objSetOption(select1, select2, select3) {
    this.select1 = select1;
    this.select2 = select2;
    this.select3 = select3;
}
//指定 3 个下拉框的值
arrText[0]= new objSetOption("选择 A:", "选择 B_1:,选择 B_2:", "选择 C_1:,选择 C_2:");
arrText[1] = new objSetOption("论文:A", "语文:A,数学:B,英语:C", "初中:A,高中:B");
arrText[2] = new objSetOption("例题:A", "显示 A2_1:值 B2_1,显示 A2_2:值 B2_2", "显示 A3_1:值
B3_1,显示 B3_2:值 B3_2");
//判断用户的选择
function select() {
    var eltSelect1 = document.form1.select1;
    var eltSelect2 = document.form1.select2;
    var eltSelect3 = document.form1.select3;
    var arrSelect1, arrSelect2, arrSelect3;
    var arrData1, arrData2, arrData3;
    //如果选择的是第一个下拉框
    with(eltSelect1) {
        var strSelect = options[selectedIndex].value;
    }
    for(i = 0;i < arrText.length;i ++) {
        arrSelect1 = arrText[i].select1;
        arrData1 = arrSelect1.split(":");
        if (arrData1[1] == strSelect) {
            arrSelect2 = (arrText[i].select2).split(",");
            for(j = 0;j < arrSelect2.length;j++) {
                arrData2 = arrSelect2[j].split(":");
                //如果选择的是第二个下拉框
                with(eltSelect2) {
                    length = arrSelect2.length;
                    options[j].text = arrData2[0];
                    options[j].value = arrData2[1];
                }
            }
            arrSelect3 = (arrText[i].select3).split(",");
            for(j = 0;j < arrSelect3.length;j++) {
                arrData3 = arrSelect3[j].split(":");
                //如果选择的是第 3 个下拉框
                with(eltSelect3) {
                    length = arrSelect3.length;
                    options[j].text = arrData3[0];
                    options[j].value = arrData3[1]; }
            }
            break; }
    }
```

```

}
//初始化设置 3 个下拉框
function init() {
var eltSelect1 = document.form1.select1;
var eltSelect2 = document.form1.select2;
var eltSelect3 = document.form1.select3;
var arrSelect1, arrSelect2, arrSelect3;
var arrData1, arrData2, arrData3;
if (eltSelect1 != undefined && eltSelect2 != undefined && eltSelect3 != undefined) {
with(eltSelect1) {
length = arrText.length;
for(i = 0;i < arrText.length;i ++) {
arrSelect1 = arrText[i].select1;
arrData1 = arrSelect1.split(":");
options[i].text = arrData1[0];
options[i].value = arrData1[1];
}
}
with(eltSelect2) {
arrSelect2 = (arrText[0].select2).split(",");
length = arrSelect2.length;
for(i = 0;i < length;i ++) {
arrData2 = arrSelect2[i].split(":");
options[i].text = arrData2[0];
options[i].value = arrData2[1];
}
}
with(eltSelect3) {
arrSelect3 = (arrText[0].select3).split(",");
length = arrSelect3.length;
for(i = 0;i < length;i ++) {
arrData3 = arrSelect3[i].split(":");
options[i].text = arrData3[0];
options[i].value = arrData3[1]; } }
}
}
init();
</script>
</body>
</html>

```

【运行效果】

三级联动菜单的效果如图 11-17 所示。

【难点剖析】

本例的重点是如何动态填充二级和三级菜单。当使用“options[selectedIndex].value”知道用户选择的是第一级菜单时，使用



图 11-17 三级联动菜单的效果



“arrSelect2 = (arrText[i].select2).split(",");” 为二级菜单填充内容；当选择的是二级菜单时，使用
“arrData3 = arrSelect3[j].split(":");” 为三级菜单填充内容。

11.18 树型目录菜单

【实例描述】

网络菜单有很多种形式，如弹出式、下拉式等。本例介绍树型菜单的制作方法。

【实现代码】

```
<script language="JavaScript">
//判断浏览器的变量
NS4 = (document.layers) ? 1 : 0;
IE4 = (document.all) ? 1 : 0;
ver4 = (NS4 || IE4) ? 1 : 0;
//定义各个层的位置及显示状态
if (ver4) {
    with (document) {
        write("<STYLE TYPE='text/css'>");
        if (NS4) {
            write(".parent {position:absolute; visibility:visible}");
            write(".child {position:absolute; visibility:visible}");
            write(".regular {position:absolute; visibility:visible}")
        }
        else {
            write(".child {display:none}")
        }
        write("</STYLE>");
    }
}
//当菜单打开时，菜单以下的东西的位置顺序往下推，菜单合起时，菜单以下的东西自动上移
function arrange() {
    nextY = document.layers[firstInd].pageY +document.layers[firstInd]. document.
height;

    for (i=firstInd+1; i<document.layers.length; i++) {
        whichele = document.layers[i];
        if (whichele.visibility != "hide") {
            whichele.pageY = nextY;
            nextY += whichele.document.height;
        }
    }
}
//初始化菜单
function initIt(){
    if (!ver4) return;
    if (NS4) {
        for (i=0; i<document.layers.length; i++) {
```



```

        whichEle = document.layers[i];
        if (whichEle.id.indexOf("Child") != -1) whichEle.visibility = "hide";
    }
    arrange();
}
else {
    divColl = document.all.tags("DIV");
    for (i=0; i<divColl.length; i++) {
        whichEle = divColl[i];
        if (whichEle.className == "child") whichEle.style.display = "none";
    }
}
}
//展开菜单的方法
function expandIt(ele) {
    if (!ver4) return;
    if (IE4) {
        whichEle = eval(ele + "Child");
        if (whichEle.style.display == "none") {
            whichEle.style.display = "block";
        }
        else {
            whichEle.style.display = "none";
        }
    }
    else {
        whichEle = eval("document." + ele + "Child");
        if (whichEle.visibility == "hide") {
            whichEle.visibility = "show";
        }
        else {
            whichEle.visibility = "hide";
        }
        arrange();
    }
}
onload = initIt;

```

需要在 body 中设计菜单的内容，代码请参考随书光盘。

【运行效果】

树型菜单效果如图 11-18 所示。

【难点剖析】

本例中因为没有使用图标，所以显示效果并不是很理想，在实际应用中，可以在节点前使用文件夹图标。本例的重点是对象的“visibility”属性，其值为“show”时显示子节点，为“hide”时隐藏节点。

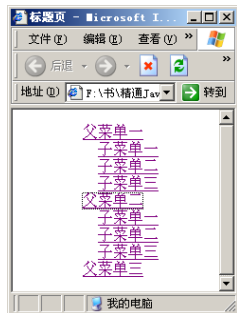


图 11-18 树型菜单效果

第 12 章

窗口特效与弹出式警告

本章导读

在网站上可以打开新的页面，通常把打开的页面称为窗口。打开窗口有多种方法，最常用的就是使用“`window.open`”。本章将详细讲解各种弹出窗口的技巧，同时介绍一些警告对话框的应用技巧。

12.1 无关闭按钮的窗口

【实例描述】

没有关闭按钮的窗口一般出现在网站说明中，用于表达网站的立场（如对一些版权问题进
行说明）。为了让用户看完说明，故不提供此页面的关闭按钮。本例学习如何制作这样的窗口。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function openLicense()
{
    //打开窗口
    window.open("http://www.google.com", "我爱的搜索", "fullscreen=7");
}
</script>
</head>
<body>
<input type=button value="打开说明" onClick="openLicense()">
</body>
</html>
```

【运行效果】

无关闭按钮的窗口如图 12-1 所示。



图 12-1 无关闭按钮的窗口

【难点剖析】

“window.open”方法用来打开新的窗口，其中没有关闭按钮的窗口样式为“fullscreen=7”。



如果要关闭此窗口，需要按“Alt+F4”组合键。

12.2 鼠标控制窗口开关

【实例描述】

鼠标移到链接上时，会自动打开一个窗口，当鼠标移走时，打开的窗口会自动关闭。本例学习利用鼠标事件控制窗口开关。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<SCRIPT Language="JavaScript">
function winopen()    //打开窗口的方法
{
    //注意新窗口的名称
    msg=open("http://www.baidu.com","baidu","toolbar=no,location=no,directories= no,
status=no,menubar=no,scrollbars=no,resizable=no,copyhistory=no,width=200,height=250");
}
</script>
<a href="" onMouseOver="winopen();" onMouseOut="msg.close();">看看窗口的变化</a>
</body>
</html>
```

【难点剖析】

本例的重点是鼠标的两个事件和操作窗口的方法。当鼠标移动到链接处时，会触发“onMouseOver”事件，当鼠标移走时，会触发“onMouseOut”事件。JavaScript的“window”对象提供一个“open”方法，用来打开新的窗口。“msg=open(...)”代码表示用“msg”代表新打开的窗口，这样在关闭窗口时才可以使用“msg.close()”方法关闭指定的窗口。

12.3 使窗口只在第一次访问时弹出

【实例描述】

第一次访问网站时，出现一个欢迎界面，第二次访问网站时便不再出现此界面。本例学习如何使窗口只弹出一次。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
```

```

//弹出窗口的方法
function openpopup(){
    window.open("http://www.baidu.com","", "width=300,height=300") //设置打开窗口
}
function get_cookie(Name)
{
    var search = Name + "="
    var returnvalue = "";
    if (document.cookie.length > 0) {
        offset = document.cookie.indexOf(search)
        if (offset != -1) { //如果指定的 Cookie 已经存在
            offset += search.length //长度指定到 Cookie 值的位置
            end = document.cookie.indexOf(";", offset); //判断是否还包括其他 Cookie 值
            if (end == -1) //如果不包括
                end = document.cookie.length; //获取 Cookie 的长度
            returnvalue=unescape(document.cookie.substring(offset, end)) //获取 Cookie 的值
        }
    }
    return returnvalue;
}
function loadpopup(){
    if (get_cookie("popped")== "") { //判断是否已经弹出过窗口
        openpopup() //如果没有则弹出窗口
        document.cookie="popped=yes" //设置 Cookie 值
    }
}
</script>
</head>
<body>
<input type=button name="btn1" value="打开窗口!" onclick="loadpopup()">
</body>
</html>

```

【难点剖析】

本例的重点是实现窗口只弹出一次的原理。当第一次弹出窗口时，将一个变量“popped”保存到 Cookie 中，如果用户第二次访问网站，判断是否包含此 Cookie，如果已经包含，则不再打开窗口。

12.4 禁止弹出警告框

【实例描述】

警告框是很多用户不喜欢的提示。本例学习如何通过重写方法，实现禁止警告框的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```



```
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
window.alert = function(str) {
    return ;
}
    alert("看看能不能弹出警示框");
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是对于“alert”方法的重写。注意重写的语法为“旧方法=function(参数){新方法的内容}”。返回值为“false”，表示不执行任何操作。

12.5 关闭窗口不提示的方法

【实例描述】

如果使用“window”对象自带的“close”方法关闭窗口，在关闭窗口时会出现一个提示形式的对话框。为了保证一些程序的运行，有时候不能弹出此提示。本例学习如何屏蔽这个关闭时的弹出窗口。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script>
function winClose()
{
    window.opener=null;           //创建当前窗口的对象为空
    window.close();               //关闭窗口
}
</script>
</head>
<body>
<input type="text" name="txt1" value="this is test!">
<input type="button" value="转换文本" onClick="javascript:winClose()">
</body>
</html>
```

【难点剖析】

本例的重点是“window.opener”的用法，其主要作用是返回创建当前窗口的对象。如果在 a 窗口中打开了 b 窗口，则 a 窗口就是创建 b 窗口的对象，可以用“window.opener”获取 a 窗口。

12.6 关闭窗口时的提示

【实例描述】

为了保证用户的操作，在关闭窗口时可以弹出提示对话框，让用户确认操作。本例学习如何实现关闭窗口的提示功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
    function ConfirmClose()
    {
        if(confirm("是否退出系统?"))                //弹出提示对话框
        {
            return true;                                //确认操作
        }
        return false;                                //取消操作
    }
</script>
</head>
<body onbeforeunload="javascript:if(ConfirmClose()){return false;}else{return true}">
</body>
</html>
```

【运行效果】

退出时的提示效果如图 12-2 所示。

【难点剖析】

本例的重点是“onbeforeunload”事件。此事件在关闭窗口时触发，可以提示用户在窗口关闭前做一些保存操作。



图 12-2 退出时的提示效果

12.7 定时弹出窗口

【实例描述】

定时弹出窗口可以起到提醒作用，如玩游戏超过一小时后，弹出一个页面提醒用户已超时。本例学习如何定时弹出指定的窗口。

【实现代码】

```
<script language="javascript">
function mywin()
{
    //弹出页面，注意页面的样式
```



```
window.open("www.google.com","我喜欢的搜索","width=300,height=100,border=0");  
}  
setTimeout("mywin()",1000);    //定时器  
</script>
```

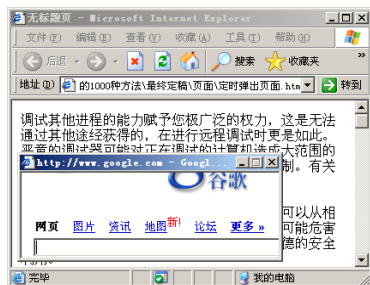


图 12-3 弹出窗口的效果

【运行效果】

弹出窗口的效果如图 12-3 所示，注意弹出的窗口没有工具栏和菜单栏等快捷操作。

【难点剖析】

本例中使用了定时器，默认时间是一秒，重点是弹出页面的“window.open”方法。其包括 3 个参数，第一个参数表示要打开窗口的地址，第二个是一般性标题，第三个是打开窗口的样式，如是否有边框、是否带滚动条、页面的高度和宽度等。

12.8 调整窗口的大小

【实例描述】

扩散是经常出现在 flash 中的效果。本例学习如何使用 JavaScript 实现扩散效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
    <title>标题页</title>  
</head>  
<body>  
<input type="button" value="调节到宽 200 高 300" onclick="window.resizeTo(300,500)">  
</body>  
</html>
```

本例的运行效果不再给出图例。

【难点剖析】

实现窗口调整的方法是“resizeTo”，其包括两个参数：宽度和高度。

12.9 打开的窗口居中

【实例描述】

很多网站在用户登录时弹出一个居中的窗口，用户填写登录信息后，再转到网站的首页。本例学习如何让打开的窗口自动居中。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```



```

<head>
<title>标题页</title>
<script language="javascript">
//参数-url 表示要打开的网站, winname 表示打开后的窗体名称
//参数 windth 表示打开窗体的宽度, height 表示打开窗体的高度
function openwindow( url,winName,width,height)
{
    xposition=0; yposition=0;
    if ((parseInt(navigator.appVersion) >= 4 ))
    {
        xposition = (screen.width - width) / 2;           //窗体居中的 x 坐标
        yposition = (screen.height - height) / 2;         //窗体居中的 y 坐标
    }
    theproperty= "width=" + width + ","                //打开窗口的属性
    + "height=" + height + ","
    + "location=0,"
    + "menubar=0,"
    + "resizable=1,"
    + "scrollbars=0,"
    + "status=0,"
    + "titlebar=0,"
    + "toolbar=0,"
    + "hotkeys=0,"
    + "screenx=" + xposition + ","                      //仅适用于 Netscape
    + "screeny=" + yposition + ","                      //仅适用于 Netscape
    + "left=" + xposition + ","                          //IE
    + "top=" + yposition;                                //IE
    window.open( url,winName,theproperty );             //打开窗口
}
</script>
</head>
<body>
<a href="javascript:openwindow('http://www.google.com','openwin',300,300)">
打开窗口</a></body>
</html>

```

【难点剖析】

本例的重点是设置打开窗口的一些属性。“window.open”方法用来打开窗口，此方法的三个参数分别代表：要打开窗口的 URL、打开窗口的名字、打开窗口的一些属性（如是否有滚动条、最大化按钮等）。本例使用“(screen.width-width)/2”和“(screen.height-height)/2”获取窗体居中的坐标点，然后指定窗体的“left”和“top”属性。

12.10 打开窗口的等待提示

【实例描述】

在打开某窗口时，为了提示用户操作正在进行，一般会给出界面提示。本例学习一个简单

的等待提示功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function openSearch(){
window.open('http://www.google.com','search')           //打开新的窗口
}
setTimeout("openSearch()",1000)                          //每隔 1 秒执行指定的方法
</script>
</head>
<body>
正在下载文件，请耐心等待.....
</body>
</html>
```

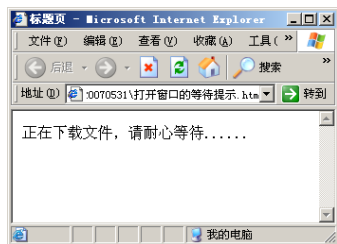


图 12-4 窗口的等待提示效果

【运行效果】

窗口的等待提示效果如图 12-4 所示。

【难点剖析】

本例使用“window.open”打开一个新的窗口，并在界面中显示了一段提示等待的文字，这种方式比较简单，并且文字一直显示在屏幕上。如果要深入应用，还可以让这段文字在打开窗口后自动消失。

12.11 在打开的窗口中返回数据

【实例描述】

当打开窗口后，客户端已经与服务器端失去了联系。那么如何从新窗口返回一个计算结果给父窗口呢？本例学习如何从打开的窗口中返回数据。

【实现代码】

父窗体的代码如下所示：

```
<HTML>
<head>
<title>无标题</title>
</head>
<BODY>
<body>
<script language="javascript">
//打开模式窗口
str =window.showModalDialog("HTMLPage4.htm","dialogWidth=200px;dialogHeight= 100px");
alert(str);//输出返回值
```

```

</script>
</body>
</HTML>

```

模式窗口中的代码如下所示：

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
  <script language=javascript>
    window.returnValue="测试的返回数据";
  </script>
</head>
<body>
</body>
</html>

```

【难点剖析】

本例的难点是如何捕获新窗口的返回值。首先使用“showModalDialog”方法打开新窗口，并使用“str”变量获取返回值。在模式窗口中，使用“returnValue”返回当前窗体中的数据。

12.12 创建弹出窗口

【实例描述】

为了提示用户操作，有时候需要弹出窗口显示提示信息，可以使用 div 层实现。本例学习使用 JavaScript 实现弹出窗口式的提示功能。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
  var oPopup = window.createPopup();           //弹出窗口对象
  var oPopupBody = oPopup.document.body;       //弹出窗口对象的文档对象
  oPopupBody.innerHTML = "这里是一个<strong>弹出窗口</strong>"; //文档内容
  oPopup.show(200, 100, 230, 50, document.body); //指定大小的窗口，显示文本内容
</script>
</head>
<body>
</body>
</html>

```

【运行效果】

弹出窗口的效果如图 12-5 所示。



图 12-5 弹出窗口的效果

【实例描述】

为了保持窗口的大小，使主要内容在窗口打开时能显示在当前界面，通常会去掉窗口的水平和垂直滚动条。本例学习如何屏蔽窗口的滚动条。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body style="overflow-x:hidden;overflow-y:hidden">
<input type="text" name="txt1" value="this is test!">
<input type="button" value="转换文本">
把页面拉伸到最小，看是否能够出现滚动条
</body>
</html>
```

【运行效果】

没有滚动条的页面如图 12-6 所示。

【难点剖析】

本例的重点是 body 的“style”属性。“style”属性可以用来为窗体设置一些特殊属性，如背景色、文本大小等。“overflow-x”和“overflow-y”是样式中用来设置水平和垂直滚动条的属性，当设置这两个属性的值为“hidden”时，窗体中再也不会出现滚动条。



图 12-6 没有滚动条的页面

12.14 页面打开的同时打开另外两个窗口

【实例描述】

如果在打开页面的同时，还需要同时打开两个窗口，如两个广告窗口，可以使用本例提供的方法。

【实现代码】

```
<html>
<head>
<title>打开新窗体</title>
```

```

<script>
window.open("http://www.google.cn","_blank");
window.open("http://www.baidu.com","_blank");
</script>
</head>
<body>
</body>
</html>

```

【难点剖析】

“window.open”方法用于在新的窗口中打开页面。本例将两个方法直接添加在“script”脚本中。当前页面加载的时候，会自动执行这两行代码，不需要触发事件。

12.15 慢慢变大的窗口

【实例描述】

窗口打开时可以使用一些特效，如卷帘式、默认最大化等。本例通过 window.resizeTo 方法实现创建窗口逐渐变大的效果。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
    var Windowsheight=100;
    var Windowswidth=100;
    var numx=5;
    function openwindow(thelocation){
        temploc=thelocation;
        if (!(window.resizeTo&&document.all)&&!(window.resizeTo&&document.getElementById))
        {
            window.open(thelocation);           //打开新窗口
            return ; }
        window.size=window.open("", "", "scrollbars");
        window.size.moveTo(0,0);                 //移动窗口到(0,0)坐标处
        window.size.resizeTo(100,100);          //改变窗口大小
        tenumxt();
    }
    function tenumxt(){
        if (Windowsheight>=screen.availHeight-3) //高度超过窗体最大高度时
            numx=0 ;
        window.size.resizeBy(5,numx);
        Windowsheight+=5;                        //高度逐次加 5
        Windowswidth+=5;                         //宽度逐次加 5
        if (Windowswidth>=screen.width-5) {      //宽度超过窗体最大宽度时

```



```
        window.location=temploc
        Windowheight=100
        Windowwidth=100
        numx=5
        return
    }
    setTimeout("tenumxt()",50)           //通过计时器，逐渐变大窗口
}
</script>
</head>
<body>
<a href="javascript:openwindow('http://www.google.com')">进入搜索</a>
</body>
</html>
```

【难点剖析】

本例的重点是 window 对象的方法：“moveTo”和“resizeTo”。“moveTo”表示移动窗口到指定位置，其包含两个参数，分别代表目标点的 x 和 y 坐标。“resizeTo”表示改变窗口的大小，其包含两个参数，分别代表窗口的高度和宽度。

12.16 设置新打开的窗口为活动窗口

【实例描述】

有时打开窗口是为了让用户输入数据，这时需要将新窗口展示在所有窗口的前面，方便用户的输入。这就是本节要实现的效果，设置新窗口为当前活动窗口。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
function winOpen()
{
    var newWin=window.open("设置新打开的窗口为活动窗口1.htm");           //打开新窗口
    newWin.focus();                                                         //窗口获得焦点
}
</script>
</head>
<body>
    <input id="Button1" type="button" value="打开" onclick="winOpen()" />
</body>
</html>
```

【难点剖析】

本例的重点是如何设置窗口为活动窗口，也可以说是当前窗口。focus 表示当前焦点，“窗

体名称.focus”则表示指定某窗体获得焦点，也就是指定此窗体为活动窗体。

12.17 页面随窗口的改变而改变

【实例描述】

当窗口改变时，页面随着窗口的改变而改变，这是页面设计的关键。本例学习如何实现这种设计。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style>body{margin:0px;}table{border-collapse:collapse;}</style>
</head>
<body>
<table width="100%" height="100%" border="3" bgcolor="gray" bordercolor="red">
    <tr>
        <td>测试页面的改变 1</td><td>测试页面的改变 2</td>
    </tr>
</table></body>
</html>
```

【难点剖析】

本例的重点是表格的宽度和高度的设置。将宽度和高度设置为百分比的形式，表示表格占当前窗口的比例。如果设置高度为 80%，则不管窗口如何改变，表格的高度永远占窗口的 80%，这样就实现了表格随窗口改变而改变的功能。

12.18 幻灯片式弹出窗口

【实例描述】

弹出窗口有很多比较有特色的效果。本例演示幻灯片方式的弹出效果。

【实现代码】

```
<script language="javascript">
var popwindow; //弹出窗口的 ID
var popwindowwidth=300; //弹出窗体的宽度
var popwindowheight=300 //弹出窗体的长度
var popwindowtop=20 ; //窗体距离屏幕顶端的距离
var popwindowURL="幻灯片式弹出窗口 1.htm"; //弹出窗体的名称
var waitingtime=4; // 窗体的停留时间
// 配置弹出窗口的速度
var pause=20;
var step=40;
```



```
var popwindowleft=-popwindowwidth-50;
var marginright;
var pagecenter;
var timer;
waitingtime= waitingtime*1000;
//显示窗口
function showWindow()
{
    popwindow = window.open(popwindowURL, "popwindow", "toolbar=no,width= "+popwindowwidth+",
height="+popwindowheight+",top="+popwindowntop+",left="+ (popwindowwidth)+"");
    marginright = screen.width+50;
    pagecenter=Math.floor(marginright/2)-Math.floor(popwindowwidth/2);
    movewindow();
}
//移动窗口
function movewindow()
{
    if (popwindowleft<=pagecenter) {
        popwindow.moveTo(popwindowleft,popwindowntop);
        popwindowleft+=step;
        timer= setTimeout("movewindow()",pause);
    }
    else {
        clearTimeout(timer);
        timer= setTimeout("movewindow2()",waitingtime);
    }
}
//移动窗口
function movewindow2()
{
    if (popwindowleft<=marginright) {
        popwindow.moveTo(popwindowleft,popwindowntop);
        popwindowleft+=step;
        timer= setTimeout("movewindow2()",pause);
    }
    else {
        clearTimeout(timer);
        popwindow.close();}
}
</script>
```

需要在 body 中添加一个按钮，代码如下所示：

```
<input type="button" id="btn1" value="打开窗口" onclick="showWindow()" />
```

【运行效果】

弹出窗口的效果如图 12-7 所示。

【难点剖析】

本例主要分两步实现弹出窗口的特效。第一步使用“window.open”方法打开窗口，第二步使用“movewindow”方法移动窗口，以实现幻灯片特效。

12.19 弹出窗口生成器

【实例描述】

IE 中弹出窗口有很多方法，最常用的是“window.open”方法。本例制作一个自动生成弹出窗口代码的工具。

【实现代码】

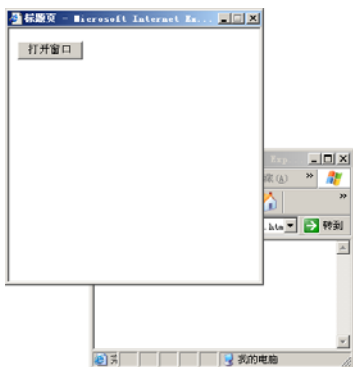


图 12-7 弹出窗口的效果

```
<STYLE type="text/css">
<!--
BODY {font-family:"宋体"; font-size: 9pt; margin-top: 0px}
.pt9 { font-family: "宋体"; font-size: 9pt}
td { font-size: 9pt }
-->
</STYLE>
<script Language="JavaScript">
function popup(url, name, width, height)
{
settings=
"toolbar=yes,location=yes,directories=yes,"+
"status=no,menubar=no,scrollbars=yes,"+
"resizable=yes,width="+width+",height="+height;
MyNewWindow=window.open(" "+url,name,settings);
}
function testmywindow()
{
result=new Array(7);
for (i=1; i<=7; i++){result[i]="no"};
with (document.detectform)
{
//以下是获取 open 方法的参数，具体某个参数取决于 body 中的控件
if (R1[0].checked) {result[1]="yes"};
if (R2[0].checked) {result[2]="yes"};
if (R3[0].checked) {result[3]="yes"};
if (R4[0].checked) {result[4]="yes"};
if (R5[0].checked) {result[5]="yes"};
if (R6[0].checked) {result[6]="yes"};
if (R7[0].checked) {result[7]="yes"};
//将参数连接成字符串
settings="toolbar="+result[1]+",location="+result[2]+",directories="+result[3]+",
status="+result[4]+",menubar="+result[5]+",scrollbars="+result[6]+",resizable="+result
[7]+",width="+newwidth.value+",height="+newheight.value;
```

```
code.value="\<a href=\"#\" onClick=\"MyWindow=window.open(\"'+url.value+'',
\"MyWindow\", \"'+settings+'\")\"\">这里是打开窗口的代码</a>";
}
}
</script>
```

【运行效果】
弹出窗口生成器的效果如图 12-8 所示。

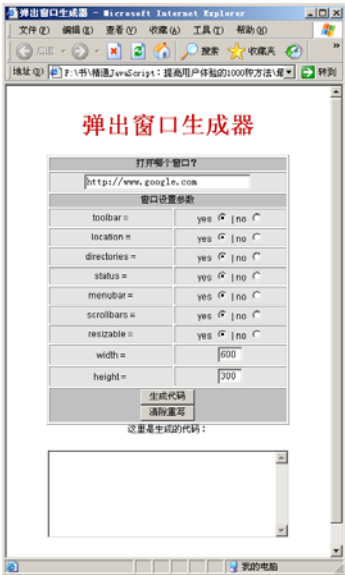


图 12-8 弹出窗口生成器的效果

【难点剖析】
本例的重点在于“window.open”方法到底有多少个参数，这些参数通过“yes/no”来开关，具体参数的意义可参考 body 中的按钮文本。

12.20 关不掉的警告框

【实例描述】
本例中的警告框在单击“确定”按钮后，仍然无法关闭。此功能很少出现在网站上，仅作参考。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
```

```
<input type=button value=单击我 onClick="var e=1;while(1==1){alert('真的关不掉吗!!!')}">
</body>
</html>
```

【运行效果】

警告效果如图 12-9 所示。

【难点剖析】

本例的重点是“while(1==1)”语句。“while”表示如果符合条件，则进入循环语句。由于条件“1==1”永远成立，故循环永远不会终止。

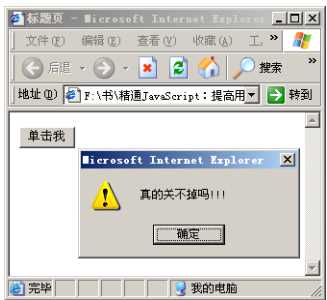


图 12-9 警告效果

12.21 循环的警告框

【实例描述】

当用户操作失误时，循环弹出一系列窗口，提示用户的操作非法。这种循环警告框在实际中用途不多，仅作参考。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script>
function circle()
{
alert(' 你 ');
alert(' 真的');
alert(' 不怕');
alert(' 这种');
alert(' 关闭');
alert(' 窗口');
alert(' 功能');
alert(' 吗? ');
return circle()
}
</script>
<a href="javascript:circle()" onmouseover="window.status='按了就要小心!';return true">
<b>最好别按</b></a>
</body>
</html>
```

【运行效果】

循环警告的提示效果如图 12-10 所示。

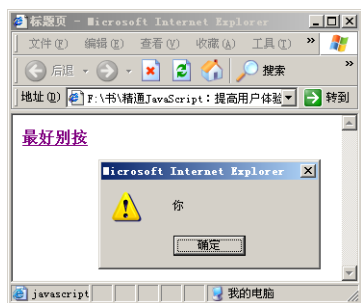


图 12-10 循环警告的提示效果

【难点剖析】

本例的重点是“circle”方法。此方法先使用“alert”方法弹出一系列警告框，然后在方法的最后，又调用了一次“circle”方法，实现此方法的循环。

12.22 屏蔽状态栏的错误提示

【实例描述】

当页面加载错误时，在状态栏会以一个黄色的叹号标识作为提示。为了不让用户注意到页面加载错误，本例介绍如何屏蔽这种错误提示。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script>
window.onerror=function(){                                //错误事件重写
    throw new Error("出错了");
}
function setError(){
    throw new Error("出错了");
}
try{
    setError("mm")                                          //触发错误
}
catch(e){                                                  //捕获错误
    alert(e.description);                                  //显示错误内容
}

</script>
</head>
<body>
</body>
</html>
```

【运行效果】

本例的运行效果如图 12-11 所示。

【难点剖析】

本例中重写了窗体的错误处理事件“onerror”。“throw new Error”表示抛出新的错误，参数内容为错误提示信息。“try…catch”语句是 JavaScript 中捕获错误的常用代码。“try”后面的表达式是正常执行的代码，“catch”后面的表达式是当出现错误时要执行的代码。

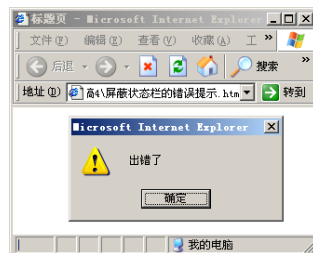


图 12-11 本例的运行效果

12.23 获取模式窗口的值

【实例描述】

在一些财务表格中，由于列数特别多，有时候需要用户打开新的窗口输入数据。本例学习如何实现在新的窗口中输入数据，并返回给父窗口的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
<script language="javascript">
function openwin()
{
  var url="获取模式窗口的值 1.htm";
  //打开模式窗口，注意模式窗口的样式
  var mydata=showModalDialog(url,null,"dialogWidth:300px;dialogHeight: 120px;
center:yes;help:No;status:no;resizable:Yes;edge:sunken");
  if(mydata)
    alert("您输入的为: " +mydata.value);
}
</script>
</head>
<body>
  <input id="Button1" type="button" value="打开窗口" onclick="openwin()" />
</body>
</html>
```

设计模式窗口“获取模式窗口的值 1.htm”的代码如下所示：

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
function ReturnWin()
{
  var returnData=new Object(); //创建变量
  returnData.value=document.getElementById("Text1").value; //设置变量的值
  window.returnValue=returnData; //窗体返回数据
  window.close(); //关闭窗口
}
</script>
</head>
<body>
  <input id="Text1" type="text" /><input id="Button1" type="button" value="返回"
onclick="ReturnWin()" />
```



```
</body>
```

```
</html>
```

【运行效果】

模式窗口的运行效果如图 12-12 所示。

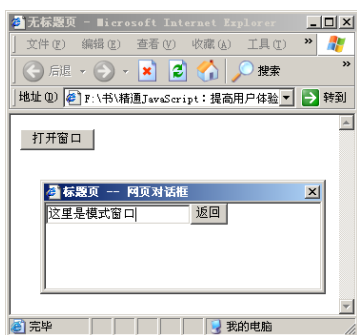


图 12-12 模式窗口的运行效果

【难点剖析】

本例的重点是如何打开模式窗口。“window.open”用来打开窗口，但用户可以不操作此窗口，而继续其他的操作。“showModalDialog”用来打开模式窗口，如果用户不关闭此窗口，则不能执行其他操作。

第 13 章

日期和时间特效

本章导读

日期和时间是 JavaScript 中常用的对象,可通过此对象判断星期、生日、纪念日等,提高网站管理的人性化。本章将详细介绍各种日期设置方法的技巧,并讲解将英文日期转化成中文日期的技巧。



13.1 指定时间关闭页面

【实例描述】

在大型门户网站中，初次打开页面时常常会同时打开一个广告页，有些网站为广告页设置了自动关闭功能，即使用户不关闭广告页，它也会在指定时间内自动关闭。本例学习如何自动关闭页面。

【实现代码】

```
<script language="javascript">
function closepage()
{
    window.close();           //关闭窗口的方法
}
setTimeout("closepage()",1000);
</script>
```



图 13-1 关闭页面时的提示界面

【运行效果】

本例指定时间为一秒，打开页面一秒后出现提示，如图 13-1 所示。系统询问是否关闭窗口，选择“是”即可。

【难点剖析】

本例中使用了定时器和窗口的关闭方法实现了定时关闭页面的功能。注意定时器的默认时间是以毫秒为单位。

13.2 最简单的时间日期特效

【实例描述】

判断当前日期为星期几，可以使用日期对象的“getDay”方法实现。本例提供一段超短代码，是显示中文星期几的方法中最短的一个。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<div id="divTime">
<script language="javascript">setInterval("divTime.innerHTML=new Date().toLocaleString()+'星期'+'日一二三四五六'.charAt(new Date().getDay())",500)
</script>
</div>
</body>
```


</html>

【运行效果】

本例的运行效果如图 13-2 所示。

【难点剖析】

本例的重点是日期对象的“getDay”方法和字符串对象的“charAt”方法。“getDay”方法返回当前日期是星期几，但是用数字来表示的，如“0”表示星期日。“charAt”方法用来截取当前文本中指定索引处的字符，而当前字符就是设置的星期“日一二三四五六”。

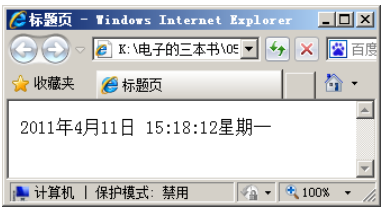


图 13-2 本例的运行效果

13.3 获取时间的最简单方法

【实例描述】

JavaScript 提供一个日期对象 Date，它提供获取时、分、秒的方法，但使用起来不够简单。本例提供一个更简单的获取时间的方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script type="text/javascript">
var now = Date().split(" ")[3];
alert(now);
</script>
</head>
<body>
</body>
</html>
```

【运行效果】

本例运行的效果如图 13-3 所示。

【难点剖析】

本例中使用字符串对象的“split”方法，将获取的日期通过空格分成 5 组。因为数组的索引是从 0 开始，所以“数组[3]”表示获取数组中第 4 个元素，这就是当前的时间。

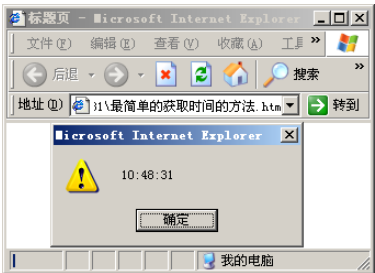


图 13-3 本例运行的效果

13.4 随日期变换的文本

【实例描述】

网站的首页通常需要显示当前的日期，为了提醒用户，可在日期后面显示一些相关信息，如



“4月11日排队日”。因为一个月最多有31天，所以可以使用数组来保存这些对应天数的信息。

【实现代码】

```
<script language="JavaScript">
    var today = new Date();                //获取当前日期
    var month = today.getMonth() + 1;      //获取月
    var date = today.getDate();             //获取日
    var year = today.getFullYear();         //获取年

    notes = new Array;
    notes[1] = "1号的信息";
    notes[2] = "2号的信息";
    notes[3] = "3号的信息";
    notes[4] = "4号的信息";
    notes[5] = "5号的信息";
    notes[6] = "6号的信息";
    notes[7] = "7号的信息";
    notes[8] = "8号的信息";
    notes[9] = "9号的信息";
    notes[10] = "10号的信息";
    notes[11] = "文明排队日";
    notes[12] = "12号的信息";
    notes[13] = "13号的信息";
    notes[14] = "14号的信息";
    notes[15] = "15号的信息";
    notes[16] = "16号的信息";
    notes[17] = "17号的信息";
    notes[18] = "18号的信息";
    notes[19] = "19号的信息";
    notes[20] = "20号的信息";
    notes[21] = "21号的信息";
    notes[22] = "无车日";
    notes[23] = "23号的信息";
    notes[24] = "24号的信息";
    notes[25] = "25号的信息";
    notes[26] = "26号的信息";
    notes[27] = "27号的信息";
    notes[28] = "28号的信息";
    notes[29] = "29号的信息";
    notes[30] = "30号的信息";
    notes[31] = "31号的信息";
    var todayMsg = notes[date];             //获取要显示的信息
</script>
```

然后在 body 元素内调用如下所示的代码，即可实现日期和信息的显示。

```
<script language="javascript">document.write(today);  document.write
(todayMsg)</script>
```

【运行效果】

随日期变换的文本运行效果如图 13-4 所示。

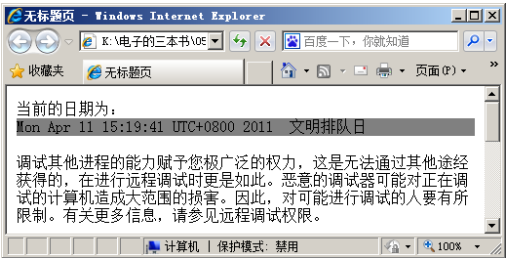


图 13-4 随日期变换的文本运行效果图

【难点剖析】

日期（Date）和数组（Array）是两个常用 JavaScript 对象，使用“new”关键字创建。读取数组时需使用索引，如“notes[date]”。

13.5 输入框的默认值为当天

【实例描述】

为了减少用户在输入一些资料创建日期时的工作量，可默认输入框的内容为当天。本例学习如何在文本框中显示当天的日期。

【实现代码】

```
<html>
<head>
<title>表格显示数据表记录</title>
</head>
<body>
<INPUT TYPE="text" id="myDate">
<SCRIPT LANGUAGE="JavaScript">
var now = new Date(); // 获取当天的日期-显示为日期和时间
myDate.value=now.getYear() + "-" + (now.getMonth()+1)+"-"+now.getDate(); // 获取年月日
</SCRIPT>
</body>
</html>
```

【运行效果】

本例的运行效果如图 13-5 所示。

【难点剖析】

本例的重点是对日期对象的操作。默认日期对象包含“年、月、日”和“时、分、秒”，如果要获取当天的日期，可使用日期

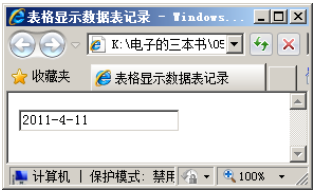


图 13-5 本例的运行效果



对象提供的获取某一部分的方法。“getYear”方法用来获取年份，“getMonth”方法用来获取月份，“getDate”方法用来获取日。

13.6 时间相加

【实例描述】

日期对象是网页中常用的对象之一，可用于倒计时、中文日期显示、日历显示等。对于一些年龄、生日等有关时间距离的计算，还需要用到日期对象的加减问题。本例学习如何计算某日期加 35 天后的日期。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script>
    var d =new Date("2007/6/12");           //创建第一个日期
    d.setDate(d.getDate()+35);              //第一个日期加 35 后，生成新的日期
    alert(d.toLocaleString());              //将日期转化为字符串
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

“new Date()”方法用来创建一个新的日期，如果括号中没有参数，则获取当前日期，否则创建一个为指定日期的日期对象。“getDate()”方法用来获取日期中的“日”。“setDate”方法用来设置当前的日期为指定日期。“toLocaleString”方法用来将日期转化为字符串。

13.7 12 小时制和 24 小时制的转换

【实例描述】

表示时间的方式有 24 小时制和 12 小时制。如果是 12 小时制的时间，通常显示“上午”或“下午”。本例学习如何实现小时制间的转换。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function timeType() {
    if (document.form.showTimeType[0].checked) { //判断选择的是哪个类型——24 小时就返回 true
```

```

        return true;
    }
    return false;
}

function showTheHours(theHour) {
    if (timeType() || (theHour > 0 && theHour < 13)) { //如果时间在 12 小时内
        return (theHour);
    }
    if (theHour == 0) { //如果时间等于 0
        return (12);
    }
    return (theHour-12); //如果时间大于 12，需要减去 12——针对 12 小时制
}

function showZeroFilled(inValue) {
    if (inValue > 9) { //设置分钟数的两位数显示，不足两位补 0
        return "" + inValue;
    }
    return "0" + inValue;
}

function showAmPm() { //显示上午或下午的方法
    if (timeType()) {
        return ("");
    }
    if (now.getHours() < 12) { //判断日期，显示 12 小时制的中文提示
        return (" 上午");
    }
    return (" 下午");
}

function showTheTime() { //显示时间的方法
    now = new Date //获取当前时间
    document.form.showTime.value = showTheHours(now.getHours()) + ":" +
showZeroFilled(now.getMinutes()) + ":" + showZeroFilled(now.getSeconds()) + showAmPm()
    setTimeout("showTheTime()",1000) //每隔 1 秒更新时间
}
</script>
</head>
<BODY onLoad="showTheTime()">
<center><form name=form>
<input type=text name=showTime size=11><p>
<input type=radio name=showTimeType checked>24 小时<br>
<input type=radio name=showTimeType>12 小时<br>
</form></center></body>
</html>

```

【运行效果】

12 小时制的效果如图 13-6 所示。

【难点剖析】

本例的难点是 12 小时制日期的显示问题。默认的显示时间是 24 小时制，如果显示 12 小时



图 13-6 12 小时制的效果

制，需要判断时间是否在 12 小时内，如果小于 12 小时，只需要原样显示时间，同时为时间添加“上午”标识；如果大于 12 小时，则需要减去 12，并添加“下午”标识。

13.8 标题栏显示时间

【实例描述】

标题栏一般显示网站名称或网页名称。本例通过时间和标题栏的调用，实现在标题栏显示时间的特效。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var clocktext;
var pagetitle = document.title;           //获取页面标题
function viewTime()
{
    today = new Date();                   //获取当前日期
    sec = today.getSeconds();             //获取秒
    hr = today.getHours();                //获取小时
    min = today.getMinutes();             //获取分钟
    if (hr <= 9) hr = "0" + hr;           //显示两位的小时
    if (min <= 9) min = "0" + min;        //显示两位的分钟
    if (sec <= 9) sec = "0" + sec;        //显示两位的秒
    var clocktext = " 当前时间是: " + hr + ":" + min + ":" + sec;
    //间隔 1000 毫秒实现时间的更新
    clocktimer = setTimeout("viewTime()", 1000);
    document.title = pagetitle + clocktext; //重新显示标题
}
if (document.all) viewTime();             //如果是 IE 浏览器
</script>
</head>
<body>
</body>
</html>
```

【运行效果】

标题栏显示时间的效果如图 13-7 所示。

【难点剖析】

本例的重点是如何显示当前时间、如何获取 IE 的标题栏。获取时间可以创建 Date 对象，但如果要一直显示当前时间，则需使用“setTimeout”定时器每隔一秒钟更新标题的时间。获取 IE 标题栏则使用“document.title”。



图 13-7 标题栏显示时间的效果

13.9 超过时间页面自动跳转

【实例描述】

本例要实现的功能主要是根据时间判断页面的跳转情况，目的是让页面完成自动跳转。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var date=new Date();

if (date.getMonth()<=2 || (date.getMonth()==2 && date.getDate()<=27))
{
    alert("跳转到第一页");
}
else if (date.getMonth()<=4 || (date.getMonth()==4 && date.getDate()<=15))
{
    alert("G 跳转到第二页");
}
else
{
    alert("跳转到第三页");
}
</SCRIPT>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例使用“new Date”创建一个日期对象，然后使用“getMonth”方法来判断当前月份，使用“getDate”方法判断当前的日期。最后根据条件，自动将用户导航到相应的页面。

13.10 分时段问候用户

【实例描述】

在网站玩游戏时，经过一段时间后会弹出提示，如玩游戏超过一小时，则提示“您已经玩了一小时了，注意休息”。本例主要是根据用户登录时间的不同，给予不同的问候。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```



```
<head>
  <title>标题页</title>
  <script language="JavaScript">
    now = new Date();
    hour = now.getHours();           //获取当前时间的小时数
    //将时间分段，并对不同提示用语
    if(hour < 6){document.write("欢迎光临，凌晨好!")}
    else if (hour < 9){document.write("欢迎光临，早上好!")}
    else if (hour < 12){document.write("欢迎光临，上午好!")}
    else if (hour < 14){document.write("欢迎光临，中午好!")}
    else if (hour < 17){document.write("欢迎光临，下午好!")}
    else if (hour < 19){document.write("欢迎光临，傍晚好!")}
    else if (hour < 22){document.write("欢迎光临，晚上好!")}
    else {document.write("欢迎光临，午夜好!")}
  </script>
</head>
<body>
</body>
</html>
```

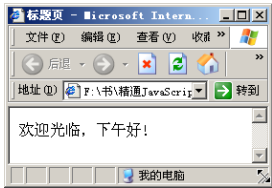


图 13-8 用户登录时的欢迎效果

【运行效果】

用户登录时的欢迎效果如图 13-8 所示。

【难点剖析】

本例的重点就是时间的获取。使用“Date”对象来获取当前时间，而“getHours”方法则是获取当前时间的小时值。

13.11 获取服务器时间

【实例描述】

有时候服务器时间与本地时间并不相同，如某些服务器可能设置在国外。本例学习如何获取服务器上的时间并显示在本地。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <script>
    var XmlHttp = new ActiveXObject("Microsoft.XmlHttp");//创建XMLHTTP对象
    XmlHttp.open("HEAD", "http://www.sohu.com", false);    //从哪个服务器上获取时间
    XmlHttp.send();                                         //连接服务器
    var offset = Date.parse(XmlHttp.getResponseHeader("Date"));//获取标头中的时间
                                                         //获取本地时间与服务器时间的间隔
    offset -= (new Date).getTime();
    function ShowTime()                                   //显示时间的方法
    {
```



```

        var d = new Date;                                // 获取当前时间
        d.setTime(d.getTime()+offset);                    // 通过服务器和本地的时间间隔获取当前服务器时间
        document.body.innerHTML=d.toLocaleString();      // 显示服务器时间
    }
    setInterval("ShowTime()", "1000");
</script>
</head>
<body>
</body>
</html>

```

【运行效果】

服务器上的时间如图 13-9 所示。

【难点剖析】

本例的重点是“offset”变量。在网页第一次运行时，使用“XMLHTTP”对象先获取服务器上的时间，并使用“offset -= (new Date).getTime()”语句计算服务器和本地时间的一个差量。获取这个差量后，就不需要每次显示时间都与服务器交互一次，而是不断执行“showTime”方法将时间显示在窗体中。

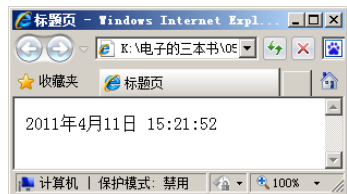


图 13-9 服务器上的时间

13.12 倒计时显示

【实例描述】

随着奥运会的临近，很多网站都显示了奥运倒计时的时间。本例就来学习如何计算当前时间与特定时间之间的距离。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
    <script language="JavaScript">
        var deadline= new Date("08/08/2012");          // 要倒计时的时间
        var symbol="2012 年 8 月 8 日";                // 提示日期标识
        var now = new Date();                            // 当前日期
        var leave =deadline.getTime() - now.getTime();   // 计算两个日期的间隔时间
        var day = Math.floor(leave / (1000 * 60 * 60 * 24)); // 间隔天数
        if (day > 0)
            document.write("今天离"+ symbol+"还有"+day +"天")
        else if (day == 0)
            document.write("只剩最后一天")
        else
            document.write("已经超过了所定时间");
    </script>
</head>
<body>

```



```
</body>
```

```
</html>
```

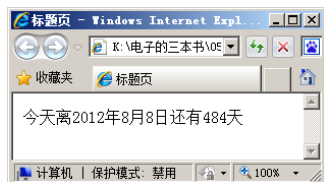


图 13-10 倒计时提示效果

【运行效果】

倒计时提示效果如图 13-10 所示。

【难点剖析】

本例的重点是如何计算两个日期的间隔。“getTime”方法返回一个整数值，此值代表了从 1970 年 1 月 1 日开始到指定日期之间时间间隔的毫秒数。本例用其计算出毫秒数后，除以“1000 * 60 * 60 * 24”后取整就得到了间隔天数。

13.13 背景时钟

【实例描述】

在页面上显示时间是网站常见的特效。本例介绍如何在页面的背景上显示时钟，以增加页面的三维效果。

【实现代码】

```
<script language=JavaScript>
function clockView()
{
    thistime= new Date(); //获取时间
    //分别获取当前时间的小时、分和秒。
    var hours=thistime.getHours();
    var minutes=thistime.getMinutes();
    var seconds=thistime.getSeconds();
    //设置时间的显示格式
    if (eval(hours) <10) {hours="0"+hours;}
    if (eval(minutes) < 10) {minutes="0"+minutes;}
    if (seconds < 10) {seconds="0"+seconds;}
    //得到最终应该显示的时间
    thistime = hours+":"+minutes+"."+seconds;
    //根据浏览器的不同设置
    if(document.all) {
        bgclocknoshade.innerHTML=thistime;
        bgclockshade.innerHTML=thistime;
    }
    if(document.layers) {
        document.bgclockshade.document.write('<div id="bgclockshade" style= "position:
absolute;visibility:visible;font-family:Verdana;color:FFAAAA;font-size:120px;top:10px;lef
ft:152px">'+thistime+'</div>');
        document.bgclocknoshade.document.write('<div id="bgclocknoshade" style= "position:
absolute;visibility:visible;font-family:Verdana;color:DDDDDD;font-size:120px;top:10px;
left:150px">'+thistime+'</div>');
    }
}
```

```
document.close();
}
var timer=setTimeout("clockView()",1000);
}
</script>
```

需要在 body 中添加三个 div，代码如下所示：

```
<body onLoad="clockView()">
<div id="bgclockshade" style="position:absolute;visibility:visible;font-family:
Arial;color:FF8888;font-size:120px;top:102px;left:152px"></div>
<div id="bgclocknoshade" style="position:absolute;visibility:visible;font-family:
Arial;color:DDDDDD;font-size:120px;top:100px;left:150px"></div>
<div id="mainbody" style="position:absolute; visibility:visible">
</div>
</body>
```

【运行效果】

背景时钟的显示效果如图 13-11 所示。

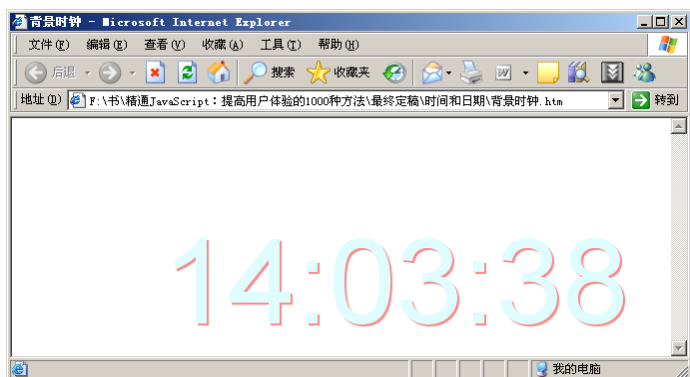


图 13-11 背景时钟的显示效果

【难点剖析】

本例的难点有两个：时间的获取与变化、三维样式显示。获取时间时，考虑到显示的美观性，需要在个位数的时间前加“0”，以转化成两位数的表示形式。因为时间是变化的，所以需要使用“setTimeout”定时器循环显示时间，注意定时器的定时单位设置为 1000 毫秒，正好等于一秒。三维样式主要依赖于 CSS 的定义，可参考代码中的 div 样式设置。

13.14 计算某天是星期几

【实例描述】

用户任意指定一个日期，程序正确地判断此日期是一周中的第几天。此功能一般用在万年历或日志查询模块中。



【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<SCRIPT LANGUAGE="LiveScript">
function checkNum(str, min, max) {                                //检测输入的数值
    if (str == "") {
        alert("请输入有效的值.");
        return false;
    }
    for (var i = 0; i < str.length; i++) {
        var ch = str.substring(i, i + 1);
        if (ch < "0" || ch > "9") {
            alert("请输入数值.");
            return false;
        }
    }
    var val = parseInt(str, 10);
    if ((val < min) || (val > max)) {
        alert("请输入数值从 1 到 "+max+".");
        return false;
    }
    return true;
}
function pushbutton(form){
    //检查年月日的有效性
    if ((checkNum(form.day.value,1,31)) && (checkNum(form.month.value, 1,12)) &&
(checkNum(form.year.value,0,2500))) {
        var cur_day = parseInt(form.day.value,10);           //获取日
        var cur_month = parseInt(form.month.value,10);       //获取月
        var cur_year = parseInt(form.year.value,10);         //获取年
    }
    getDayOfWeek(cur_year+"-"+cur_month+"-"+cur_day);        //调用判断星期几的方法
}
function getDayOfWeek(dayValue){
    var day = new Date(Date.parse(dayValue.replace(/-/g, '/')));
                                                                    //将日期值格式化
    var today = new Array("星期日","星期一","星期二","星期三","星期四","星期五","星期六"); //
创建星期数组
    alert( today[day.getDay()])                                     //返回一个星期中的某一天，其中 0 为星期日
}
</SCRIPT>
<FORM NAME = "calDay">
<PRE><B>日:</B>
<INPUT TYPE="num" name="day" onChange="if (!checkNum(this.value, 1,

```

```

31))){this.focus();this.select();} else {}" size=10 value="">
    <B>月:</B>
    <INPUT TYPE="num" name="month" onChange="if (!checkNum(this.value, 1,
12))){this.focus();this.select();} else {}" size=10 value="">
    <B>年:</B>
    <INPUT TYPE="num" name="year" onChange="if (!checkNum(this.value, 0,
2500))){this.focus();this.select();} else {}" size=10 value="">
    <INPUT TYPE="button" name="Find_Out" value="计算星期几"
onclick="pushbutton(this.form)"></PRE>
</FORM>
</body>
</html>

```

【运行效果】

显示的星期效果如图 13-12 所示。

【难点剖析】

本例的重点是“getDayOfWeek”方法，其中使用了 Date 对象的“getDay”方法来获取当前日期是一周中的第几天。此处要注意“getDay”和“getDate”方法的区别：“getDate”方法获取一月中的第几天，而“getDay”获取一周中的第几天。

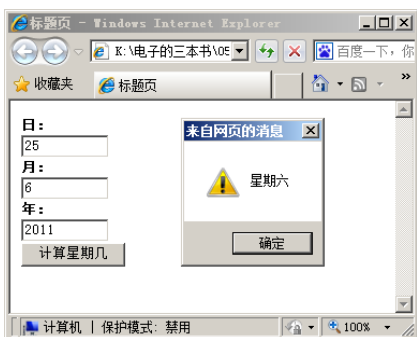


图 13-12 显示的星期效果

13.15 计算时间差

【实例描述】

如何计算两个时间之间的间隔，JavaScript 并没有提供专门的函数。本例提供一个方法，计算两个时间的时间差。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
function calTime()
{
var time1 = new Date().setHours(12,25,30);           //三个参数分别是小时、分、秒
var time2 = new Date().setHours(20,12,10);           //创建另一个时间
var timediff = time2 - time1;                         //获取两个时间差，结果为毫秒
timediff = timediff/(60*60*1000);                     //毫秒换算成小时
return timediff;
}
</script>
</head>
<body>

```



```
<input type=button value="计算" onClick="alert (calTime())">
</body>
</html>
```

【难点剖析】

本例的重点是“setHours”方法，用来生成一个标准的时间对象。“timediff/(60*60*1000)”是将一个毫秒值换算为小时值，这样可以输出两个时间之间的小时差，如果要输出毫秒差，可调整此代码。

13.16 计算用户浏览网页的时间

【实例描述】

当用户离开网页时，可以计算用户在该网页的停留时间，并以 alert 的方式进行提醒。注意是在某网页的停留时间，而不是某网站。

【实现代码】

```
<script language="Javascript">
    pageOpenTime = new Date();           //定义打开网页的时间
    function goodbye()
    {
        pageCloseTime = new Date();      //定义关闭网页的时间
        minutes = (pageCloseTime.getMinutes() - pageOpenTime.getMinutes());
        seconds = (pageCloseTime.getSeconds() - pageOpenTime.getSeconds());
        time = (seconds + (minutes * 60));
        alert('总共停留了' + time + '秒,一路走好!');
    }
</script>
```

当用户离开的时候，需要调用上面的“goodbye”方法，所以为 body 指定“onunload”事件，代码如下所示：

```
<body onunload="goodbye()">
```

【运行效果】

该特效的提示效果如图 13-13 所示。

【难点剖析】

读取系统日期和时间的是 Date 对象，当用户打开网页时，使用 Date 获取打开时间，用户关闭网页时，调用方法“goodbye”获取关闭时间，然后将分钟数换算为秒数，计算两个时间的间隔，最后用“alert”方法提醒用户。

13.17 记录页面的修改时间

【实例描述】

为了让用户知道网站的内容是否更新，可在首页上提示页面的更新时间。



图 13-13 该特效的提示效果

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="JavaScript">
document.write("本页面最后更新的时间是: " + document.lastModified + " ")
</script></body>
</html>
```

【运行效果】

页面的提示效果如图 13-14 所示。

【难点剖析】

本例的重点是 Document 对象的“lastModified”属性，其用来获取当前页面的最后更新时间。使用 Document 对象可以获取文档的一些相关属性，如创建时间、页面大小等。

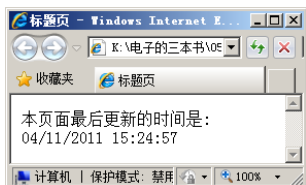


图 13-14 页面的提示效果

13.18 将日期转换为字符串的方法

【实例描述】

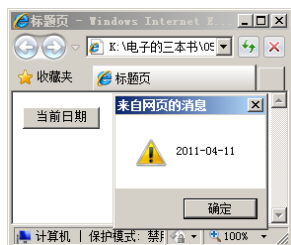
为了统一日期的格式，大部分网站的后台程序都会提供一个日期转换方法，将日期统一按照 8 位数的形式显示。本例介绍如何实现这种转换。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function getMyDate(tmpDate)
{
    var date1,date2;                                //定义两个变量
    date1 =tmpDate.getMonth()+1+"";                 //获取当前月份+1 的值
    if(date1.length<2)                               //判断当前月份是否是双位数，10 以上
        date1="0"+date1;                             //单位数的情况下，需要在月份前补 0
    date2 =tmpDate.getDate()+"";                     //获取当前日期
    if(date2.length<2)                               //判断日期的位数是否是双位
        date2="0"+date2;                             //不足双位补 0
    return tmpDate.getYear()+"-"+date1+"-"+date2;    //返回完整的日期
}
</script>
</head>
<body>
```



```
<input id="Button1" type="button" value="当前日期" onclick="alert(getMyDate(new  
Date()))" />  
</body>  
</html>
```



【运行效果】

转换后的日期如图 13-15 所示。

【难点剖析】

本例的重点是判断日期中月份的位数，如果只有一位，则需要在前补“0”。获取当前日期的函数是“getDate”，获取月份的函数是“getMonth”，获取年份的函数是“getYear”。

图 13-15 转换后的日期

13.19 检测是否是闰年

【实例描述】

在判断日期时需要注意闰年的二月份天数为 29 天，非闰年为 28 天。本例学习一个判断闰年的方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
<script LANGUAGE="JavaScript">  
function checkYear(year)  
{  
    //判断是否能被 4 整除且不能被 100 整除，还有一个是能被 400 整除-闰年的标准  
    return ((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0) ? 1 : 0;  
}  
function Judge(form)  
{  
    year = form.year.value;  
    var Check1 = parseFloat(year); //获取由字符串转换成的浮点数  
    for (var i = 0; i < year.length; i++) { //逐字符判断年份的有效性  
        var sLetterCheck1 = year.substring(i, i+1);  
        if (sLetterCheck1 < "0" || sLetterCheck1 > "9") {  
            alert("请输入一个有效的年份");  
            form.year.focus();  
            form.year.select();  
            return;  
        }  
    }  
}  
if (year < 1582) {  
    form.result.value = "";  
    alert("对不起，你输入的年份必须大于 1581。");  
}
```



```

        form.year.focus();
        form.year.select();
        return;
    }
    checkYear(year); //判断年份是否是闰年
    if (!checkYear(year)) form.result.value = "不是闰年";
    else form.result.value = "是闰年";
}
</script>
</head>
<body>
    <form>
        检测某一年是否是闰年:<br>
        年份:
        <input type="text" name="year" size="4">
        <input type="button" value="检测是否是闰年" onClick="Judge(this.form);" name=
"button">
        <input type="text" name="result" size="15">
    </form>
</body>
</html>

```

【运行效果】

检测的效果如图 13-16 所示。

【难点剖析】

本例的重点是闰年的判断标准。判断一个年份是否闰年，一是判断其是否能被 4 整除，同时还不能被 100 整除（如 2100 就不是闰年）；二是判断此年份是否能被 400 整除，如果能则一定是闰年。



图 13-16 检测的效果

13.20 年份加减函数

【实例描述】

年份加减功能经常出现在财务软件中。本例使用 JavaScript 实现年份加减函数，可在网页中直接调用。

【实现代码】

```

<html>
<head>
<SCRIPT LANGUAGE="JavaScript">
    var d=new Date() //获取当天日期
    document.write("<select id=selyear>"); //输出一个下拉框
    var int;
    int=new Array() //创建数组

```



```
for(i=2003;i<2013;i++){ //依次添加年份到数组
    int[i]=i;
    document.write("<option "+((d.getYear()==i)?"selected":"" )+" value=" + int[i] + ">"
+ int[i] + "</option>"); //默认选中当前年份
}
document.write("</select>年");
//实现年份加减的方法
function DateAdd(n){
    var dlt = document.getElementById("selyear"); //获取年份下拉框
    var len = dlt.options.length; //获取有多少个年份
    var idx = dlt.selectedIndex - n ; //年份的索引——默认当前年
    dlt.selectedIndex = idx < 0 ? 0 : (idx > len-1 ? len-1 : idx); //选定年份
}
</SCRIPT>
<title>无标题</title>
</head>

<body>
<form name="form1" method="post" action="">
    <input type="button" id="YearPre" name="YearPre" value="上一年" class="button" onclick=
"DateAdd(1);" />
    <input type="button" id="YearNext" name="YearNext" value="下一年" class="button" onclick=
"DateAdd(-1);" />

</form>
</body>
</html>
```



图 13-17 本例的运行效果

【运行效果】

本例的运行效果如图 13-17 所示。

【难点剖析】

本例的重点是如何将年份动态添加到下拉框中，并指定当前年份。默认选中的年份是当前年份。通过一个循环往下拉框中添加“option”项，然后通过“dlt.selectedIndex - n”获取指定索引处的年份，参数“n”可以是正值，也可以是负值。

13.21 精确到千分之一秒

【实例描述】

JavaScript 中的日期对象只能显示到秒。本例学习如何显示更精确的时间，类似于体育运动中常用的跑表。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```

<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
var ms = 0;
var state = 0;
function startstop()
{
    if (state == 0) {                                //开始走秒
        state = 1;
        then = new Date();                            //获取当前时间
        then.setTime(then.getTime() - ms);
    }
    else {                                            //结束走秒
        state = 0;
        now = new Date();
        ms = now.getTime() - then.getTime();
        document.form1.time.value = ms;
    }
}
function timeReset()                                //重置时间的方法
{
    state = 0;
    ms = 0;
    document.form1.time.value = ms;
}
function display()                                  //开始显示时间
{
    setTimeout("display();", 1);                    //设置定时器
    if (state == 1) {now = new Date();                //获取当前的新时间
    ms = now.getTime() - then.getTime();              //通过时间差计算毫秒
    document.form1.time.value = ms;                  //显示毫秒
    }
}
</SCRIPT>
</head>
<body onLoad="display()">
    <form name="form1">
        本例的时间可以准确到千分之一秒<br>
        计时:
        <INPUT TYPE="text" Name="time" /><br />
        <INPUT TYPE="BUTTON" Name="btnSet" VALUE="开始/停止" onClick="startstop()" />
        <INPUT TYPE="BUTTON" NAME="reset" VALUE="重置" onClick="timeReset()" />
    </form>
</body>
</html>

```

【难点剖析】

本例的重点是毫秒的计算。当用户单击“开始/停止”按钮时，首先判断现在是否有时间在



显示，有则表示要停止显示时间，否则开始显示时间。开始计时时，首先要获取当前时间，然后每隔一毫秒修改一下当前时间。最终显示的内容是当前时间减去初次显示时的时间，结果为毫秒值。

13.22 距离某天的时间

【实例描述】

距离某天的时间一般用于提醒特别重大的日子，如建国、结婚纪念日等。本例以建国日为标准，学习如何获取距离某天的时间。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body bgcolor="#fef4d9" onLoad="startclock()">
<SCRIPT LANGUAGE="JavaScript">
//基本变量的定义
var timerID;
var timerRunning = false;
var today = new Date();
var startday = new Date();
var secPerDay = 0;
var minPerDay = 0;
var hourPerDay = 0;
var secsLeft = 0;
var secsRound = 0;
var secsRemain = 0;
var minLeft = 0;
var minRound = 0;
var minRemain = 0;
var timeRemain = 0;

//停止定时器的方法
function stopclock()
{
    if(timerRunning)
        clearTimeout(timerID);

    timerRunning = false;
}
//开始定时器的方法
function startclock()
{
    stopclock();
```

```

        showtime1();
    }
    //计算时间的方法
    function showtime1()
    {
        startday = new Date("Oct 1, 1949 00:00 EDT"); //创建初始时间
        startday.setYear("1949"); //设置年份
        today = new Date(); //获取当前时间
        secsPerDay = 1000; //最终转换为毫秒
        minPerDay = 60 * 1000; //每分钟的毫秒
        hoursPerDay = 60 * 60 * 1000; //每小时的毫秒
        PerDay = 24 * 60 * 60 * 1000; //每天的毫秒

        /* Seconds */
        secsLeft = (today.getTime() - startday.getTime()) / minPerDay; //距今天的分钟数
        secsRound = Math.round(secsLeft); //四舍五入
        secsRemain = secsLeft - secsRound;
        secsRemain = (secsRemain < 0) ? secsRemain = 60 - ((secsRound - secsLeft) * 60) :
secsRemain = (secsLeft - secsRound) * 60;
        secsRemain = Math.round(secsRemain); //计算取分钟后剩余的秒数
        /* 分钟*/
        minLeft = ((today.getTime() - startday.getTime()) / hoursPerDay); //距今天的小时数
        minRound = Math.round(minLeft); //四舍五入
        minRemain = minLeft - minRound;
        minRemain = (minRemain < 0) ? minRemain = 60 - ((minRound - minLeft) * 60) : minRemain
= ((minLeft - minRound) * 60);
        minRemain = Math.round(minRemain - 0.495); //计算取小时后剩余的分钟数

        /* 小时 */
        hoursLeft = ((today.getTime() - startday.getTime()) / PerDay); //距今天的天数
        hoursRound = Math.round(hoursLeft); //四舍五入
        hoursRemain = hoursLeft - hoursRound;
        hoursRemain = (hoursRemain < 0) ? hoursRemain = 24 - ((hoursRound - hoursLeft) *
24) : hoursRemain = ((hoursLeft - hoursRound) * 24);
        hoursRemain = Math.round(hoursRemain - 0.5); //计算取天数后剩余的小时数

        /* 天 */
        daysLeft = ((today.getTime() - startday.getTime()) / PerDay); //距今天的天数
        daysLeft = (daysLeft - 0.5);
        daysRound = Math.round(daysLeft); //四舍五入
        daysRemain = daysRound;

        /* Time */
    }

```



```
        day_rem = " 天, "
        hour_rem = " 小时, "
        min_rem = " 分, "
        sec_rem = " 秒"

        timeRemain = daysRemain + day_rem + hoursRemain + hour_rem + minRemain + min_rem
+ secsRemain + sec_rem;

        document.up.myTxt.value = timeRemain;                //使用文本框输出内容
        timerID = setTimeout("showtime1()",1000);            //每隔 1 秒更新一下时间
        timerRunning = true;

    }

</SCRIPT>
<FORM NAME="up">
<INPUT TYPE="text" NAME="myTxt" SIZE="47" VALUE="">
</FORM>
<P>
<FONT SIZE=+1 FACE="Arial">
距离建国已经
</FONT>
</body>
</html>
```

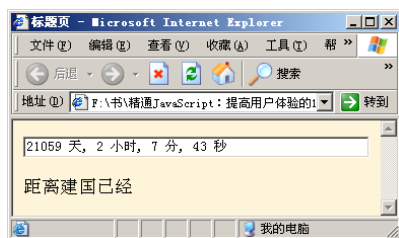


图 13-18 本例运行效果

【运行效果】

本例运行效果如图 13-18 所示。

【难点剖析】

本例的重点在于时间（天、时、分、秒）的获取。在“showtime1”方法中，使用“new Date("Oct 1, 1949 00:00 EDT")”创建了一个建国日期对象，又使用“new Date()”获取当前的日期对象。最后分别获取两个时间的天数、小时、分、秒的间隔，并在文本框中显示最后的计算结果。

13.23 判断两个字符串日期的大小

【实例描述】

日期型数据可以使用日期对象的一些方法来判断大小，如 getDate、getFullYear 等。本例将使用一个更简单的方法实现字符串型日期的判断。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function Judge()
{
```

```

        if(document.getElementById("beginTime").value<=document.getElementById("endTime").
value)
            alert("开始日期小于结束日期");
        else
            alert("开始日期大于结束日期");
    }
</script>
</head>
<body>
开始日期: <input type="text" id="beginTime"> <br/>
结束日期: <input type="text" id="endTime">
<input type="button" value="判断" onclick="Judge()" />
</body>
</html>

```

【难点剖析】

本例的重点是字符型数据默认的比较方法。字符串是逐字符进行比较的，所以在比较日期型数据时，可先将日期数据转换为字符串，然后使用字符串默认的比较方法进行比较。

13.24 显示登录时间

【实例描述】

用户登录到网站时，有一个文本框一直显示用户的停留时间。显示时间是随着当前的时间不断变化的。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<form name=forms> <font size=3>
<CENTER><div align=center><p></font>
<font color=red size=3>
您在本网停留了</font>
<font color=#80FF80><br>
<input type=text name=input1 size=10>
</font></p></div></center></font>
<SCRIPT language=javascript>
var sec=0;var min=0;var hou=0;flag=0;
window.setTimeout("update();" ,1000);
function update()
{
    sec++;
    if(sec==60){

```

//每隔 1 秒更新时间
//实现更新的主要方法
//秒数自增



```
sec=0;min+=1; //如果秒数超过 60,分钟数就自增
}
if(min==60){ //如果分钟数超过 60,小时数就自增
min=0;hou+=1;
}
if((min>0)&&(flag==0))
{
    window.alert("您停留了 1 分钟,欢迎再来!");
    flag=1;
}
document.forms.input1.value=hou+"时"+min+"分"+sec+"秒"; //在文本框中输出时间
window.setTimeout("update();" ,1000); //定时更新时间,每隔一秒更新一次
}
</SCRIPT>
</form>
</body>
</html>
```

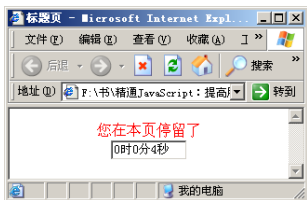


图 13-19 显示的登录时间

【运行效果】

显示的登录时间如图 13-19 所示。

【难点剖析】

本例的重点其实是“setTimeout”计时器。为了实现时间不断变化,必须使用“setTimeout”不断调用“update”方法。将定时时间设置为“1000”会每一秒更新一次显示的文本。

13.25 中文日期样式(一)

【实例描述】

由于操作系统的内容是英文的,所以日期显示一般不符合中文的显示格式。本例使用一种简单的方法将日期转换为中文样式。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
<script language="javascript">
function number(index1)
{
    //定义中文数值的显示
    var numberstring="一二三四五六七八九十";
    if(index1 ==0) {document.write("十")}
    if(index1 < 10){
        document.write(numberstring.substring(0+(index1-1),index1))}
    else if(index1 < 20 ){
        document.write("十"+numberstring.substring(0+(index1-11),(index1-10)))}
```



```

else if(index1 < 30 ){
document.write("二十"+numberstring.substring(0+(index1-21),(index1-20)))}
else{
document.write("三十"+numberstring.substring(0+(index1-31),(index1-30)))}
}
var today1 = new Date();           //获取当前日期
var month = today1.getMonth()+1;
var date = today1.getDate();
var day = today1.getDay();
document.write("<br><strong><small><center>")
document.write("二零壹壹年");
number(month);                     //将月份转换为中文
document.write("月");
number(date);                      //将日期转换为中文
document.write("日</small><center>")
</script>
</head>
<body>
</body>
</html>

```

【运行效果】

中文日期样式如图 13-20 所示。

【难点剖析】

本例的重点在于日期或月份超过 10 时中文数字的获取。因为即使日或月大于 10, 也不会超过两位数, 所以本例使用方法“substring”截取第二位, 来判断 10 以后的中文数字。

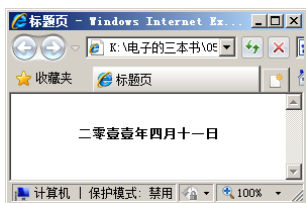


图 13-20 中文日期样式

13.26 中文日期样式（二）

【实例描述】

默认日期是以数字形式显示的, 如星期日是 0。为了表示正确的日期, 本例学习如何显示中文日期。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
    document.write("<center><font size=+1><b>") //设置在文档中的显示样式
    today = new Date()                          //定义新的日期对象
    document.write("<center><small>")           //根据 getDay 方法, 判断是星期几
    if (today.getDay() == 5) document.write("星期五")
    if (today.getDay() == 6) document.write("星期六")

```

```
if (today.getDay() == 0) document.write("星期日")
if (today.getDay() == 1) document.write("星期一")
if (today.getDay() == 2) document.write("星期二")
if (today.getDay() == 3) document.write("星期三")
if (today.getDay() == 4) document.write("星期四")
document.write("</small></center>")
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是 Date 对象的应用。获取当前日期使用“new Date()”语句。获取当天是一周中的第几天使用“getDay”方法，此方法返回的值为“0~6”，从星期日开始。

13.27 状态栏动态显示时间

【实例描述】

为了提醒用户登录网站的时间，很多网站在页面右上角显示当前时间，但考虑网站的整体效果，可将时间显示在状态栏中。本例学习如何在状态栏中显示时间。

【实现代码】

```
<script language="javascript">
function viewtime()
{
    window.setTimeout("viewtime()",1000);           //设置定时器
    today = new Date();                               //获取时间
    self.status = today.toString();                  //状态栏显示时间
}
</script>
```

需要在 body 的事件中加载显示时间的方法，代码如下所示：

```
<body onload=viewtime()>
```

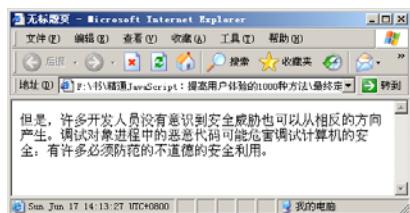


图 13-21 状态栏时间显示界面
的状态栏使用的是“self.status”。

【运行效果】

状态栏时间显示界面如图 13-21 所示。

【难点剖析】

本例中的重点有 3 个：定时器、获取时间和调用状态栏。定时器用来在指定的时间内循环执行任务，指定时间的单位是毫秒。获取时间使用的是 Date 对象。调用本窗口

13.28 页面访问时间限制

【实例描述】

有些页面设置了访问时间, 要求用户在指定的时间内保存页面。本例学习如何实现这种功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT language="JavaScript">
var limit="1:02"; //设置剩余的时间
var tmpTime=limit.split(":"); //将分和秒切割开
var myTime=tmpTime[0]*60+tmpTime[1]*1; //获取剩余的秒数
function calTime(){
    if (myTime==1)
        window.location="www.google.com" //时间到了后导航到的地址
    else{
        myTime-=1; //开始倒数时间
        curmin=Math.floor(myTime/60); //当前剩余分钟数
        cursec=myTime; //当前剩余秒数
        if (curmin!=0) //如果分钟数不为 0
            curtime="你只有"+curmin+"分钟的时间保存此页。";
        else
            curtime="你只有"+cursec+"秒的时间访问此页, 请尽快保存。";
        setTimeout("calTime()",1000); //设置定时器, 不断变化提示时间
        document.form1.txttime.value=curtime; //在文本框中显示剩余时间
    }
}
</SCRIPT>
</head>
<body bgcolor="#fabaaa" onload="calTime()">
<form name="form1">
<p><input type="text" name="txttime" size="40" />
</p>
</form>
</body>
</html>
```

【难点剖析】

本例的重点是秒数的计算。如果设置的剩余时间中有分钟数, 需要先将分钟数转换为秒数。在代码中的“var myTime=tmpTime[0]*60+tmpTime[1]*1”处要注意, 虽然“tmpTime[1]*1”和“tmpTime[1]”计算结果一样, 但为了保持变量类型一致, 还需要使用“tmpTime[1]*1”。“Math.floor(myTime/60)”用来判断是否还有分钟数。



13.29 显示英文的上、下午时间标签

【实例描述】

显示的时间信息默认是 24 小时制，如果要显示成上、下午样式的时间，就是将时间显示成 12 小时制样式。本例通过获取当前时间的小时数与 12 进行比较，以判断显示的时间是上午还是下午。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script LANGUAGE="JavaScript">
    myDate = new Date();           //创建日期对象，获取当前日期
                                   //获取日期中的月、日、年

    document.write('<font size="2" face="Arial"><B>' + (myDate.getMonth() + 1)
+ "/" + myDate.getDate() + "/" + myDate.getYear() + '</B></font><BR>');

    var Hours;
    var Mins;
    var Time;
    Hours = myDate.getHours();     //获取小时
    if (Hours >= 12) {
        Time = " P.M.";           //如果时间已经是下午（大于 12 点）则显示 p.m.
    }
    else {
        Time = " A.M.";           //如果时间小于 12 点，则显示 a.m.
    }
    if (Hours > 12) {               //如果小时数大于 12，则减 12，主要是为了显示上午和下午
        Hours -= 12;
    }
    if (Hours == 0) {              //0 点时显示 12 点
        Hours = 12;
    }
    Mins = myDate.getMinutes();   //获取当前的分钟数
    if (Mins < 10) {
        Mins = "0" + Mins;        //如果分钟数小于 10，则前面补 0
    }

    document.write('<font size="2" face="Arial"><B>' + Hours + ":" + Mins + Time +
'</B></font>');
</script>
</body>
</html>
```

【运行效果】

显示英文的上、下午时间标签如图 13-22 所示。

【难点剖析】

本章的重点是获取月、日、年、小时和分的一些方法。Date 对象是 JavaScript 的标准时间对象，其提供的时间包含“年、月、日”和“时、分、秒”。此对象也提供“getYear”、“getMonth”、“getDate”、“getHours”、“getMinutes”和“getSeconds”等方法，用来实现时间中某一部分（如分、秒等）的获取。

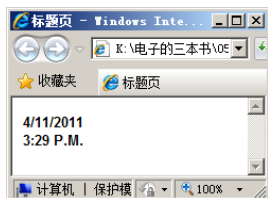


图 13-22 显示英文的上、下午时间标签

13.30 用 JavaScript 制作的特色时钟

【实例描述】

JavaScript 的功能非常强大，可结合 CSS 实现日历、时钟等。本例学习如何实现用纯脚本制作的时钟。

【实现代码】

```
<SCRIPT language=javascript>
pX=400;pY=200;
obs = new Array(13);
function ob ()
{
    for (i=0; i<13; i++) {
        if (document.all) obs[i]=new Array (eval('ob'+i).style,-100,-100)
        else obs[i] = new Array (eval('document.ob'+i),-100,-100)
    }
}
//设置时钟中各时间点的位置
function cl(a,b,c)
{
    //判断浏览器格式
    if (document.all) {
        //IE 格式
        if (a!=0) b+=-1
        eval('c'+a+'.style.pixelTop='+ (pY+(c))) //y 坐标
        eval('c'+a+'.style.pixelLeft='+ (pX+(b))) //x 坐标
    }
    else{
        //其他浏览器格式
        if (a!=0) b+=10
        eval('document.c'+a+'.top='+ (pY+(c)))
        eval('document.c'+a+'.left='+ (pX+(b)))
    }
}
if (document.all) c0.style.pixelLeft=26
```



```
}
//设置时针、分针、秒针的显示位置
function runClock()
{
    for (i=0; i<13; i++) {
        obs[i][0].left=obs[i][1]+pX
        obs[i][0].top=obs[i][2]+pY
    }
}
//设置时、分、秒的显示效果
var lastsec;
function timer()
{
    time = new Date ()
    sec = time.getSeconds() //获取当前时间的秒分
    if (sec!=lastsec) {

        lastsec = sec
        sec=Math.PI*sec/30 //秒
        min=Math.PI*time.getMinutes()/30 //分
        hr =Math.PI*((time.getHours()*60)+time.getMinutes())/360 //时
        for (i=1;i<6;i++) {
            obs[i][1] = Math.sin(sec) * (44 - (i-1)*11)-16;
            if (document.layers)obs[i][1]+=10;
            obs[i][2] = -Math.cos(sec) * (44 - (i-1)*11)-27;
        }
        for (i=6;i<10;i++) {
            obs[i][1] = Math.sin(min) * (40 - (i-6)*10)-16;
            if (document.layers)obs[i][1]+=10;
            obs[i][2] = -Math.cos(min) * (40 - (i-6)*10)-27;
        }
        for (i=10;i<13;i++) {
            obs[i][1] = Math.sin(hr) * (37 - (i-10)*11)-16;
            if (document.layers)obs[i][1]+=10;
            obs[i][2] = -Math.cos(hr) * (37 - (i-10)*11)-27;
        }
    }
}
//设置钟点
function setNum()
{
    cl (0,-67,-65);
    cl (1,10,-51);
    cl (2,28,-33);
    cl (3,35,-8);
    cl (4,28,17);
    cl (5,10,35);
    cl (6,-15,42);
```

```

    cl (7,-40,35);
    cl (8,-58,17);
    cl (9,-65,-8);
    cl (10,-58,-33);
    cl (11,-40,-51);
    cl (12,-16,-56);
}
</SCRIPT>

```

【运行效果】

特色时钟的效果如图 13-23 所示。

【难点剖析】

本例的重点是实现时钟布局的原理。代码中使用了以“c”开头的 12 个变量分别代表 12 个时钟数，然后又使用以“obs”开头的 12 个变量勾画出了时针、分针和秒针。

13.31 自定义的日历

【实例描述】

在很多网络名人的 Blog 上都增加了一些个性时尚的日历。本例学习如何使用 JavaScript 设计自己的日历。

【实现代码】

```

<script language="javascript">
    var months = new Array("一", "二", "三", "四", "五", "六", "七", "八", "九", "十", "十一", "十二");
    //定义月份
    var daysInMonth = new Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    //定义每月的天数
    var days = new Array("日", "一", "二", "三", "四", "五", "六"); //定义星期几
    var classTemp;
    var today=new getToday();
    var year=today.year; //获取年份
    var month=today.month; //获取月份
    var newCal;
    //用来获取指定年月中的天数
    function getDays(month, year)
    {
        if (1 == month) return ((0 == year % 4) && (0 != (year % 100))) || (0 == year % 400) ?
29 : 28;
        else return daysInMonth[month];
    }
    //获取今天的年、月、日
    function getToday() {
        this.now = new Date();

```

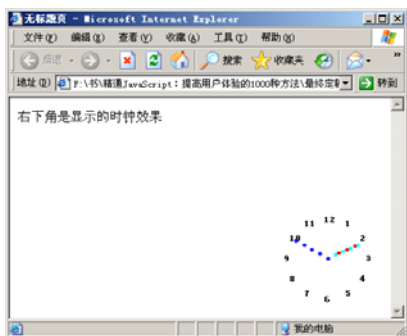


图 13-23 特色时钟的效果



```

    this.year = this.now.getFullYear();
    this.month = this.now.getMonth();
    this.day = this.now.getDate();
}
//定义日历表的显示方法
function Calendar() {
    newCal = new Date(year,month,1);
    today = new getToday();
    var day = -1;
    var startDay = newCal.getDay();
    var endDay=getDays(newCal.getMonth(), newCal.getFullYear());
    var daily = 0;
    if ((today.year == newCal.getFullYear()) &&(today.month == newCal.getMonth ()))
    {
        day = today.day;
    }
    var caltable = document.all.caltable.tBodies.calendar;
    var intDaysInMonth =getDays(newCal.getMonth(), newCal.getFullYear());

    for (var intWeek = 0; intWeek < caltable.rows.length;intWeek++)
        for (var intDay = 0;intDay < caltable.rows[intWeek].cells.length;intDay++)
        {
            var cell = caltable.rows[intWeek].cells[intDay];
            var montemp=(newCal.getMonth()+1)<10?("0"+(newCal.getMonth()+1)): (newCal.
getMonth()+1);
            if ((intDay == startDay) && (0 == daily)){ daily = 1;}
            var daytemp=daily<10?("0"+daily):(daily);
            var d="<"+newCal.getFullYear()+"-"+montemp+"-"+daytemp+">";
            if(day==daily) cell.className="DayNow";
            else if(intDay==6) cell.className = "DaySat";
            else if (intDay==0) cell.className = "DaySun";
            else cell.className="Day";
            if ((daily > 0) && (daily <= intDaysInMonth))
            {
                cell.innerText = daily;
                daily++;
            } else
            {
                cell.className="CalendarTD";
                cell.innerText = "";
            }
        }
    document.all.year.value=year;
    document.all.month.value=month+1;
}
//实现月份向前翻页的方法
function subMonth()
{

```



```

    if ((month-1)<0)
    {
        month=11;
        year=year-1;
    } else
    {
        month=month-1;
    }
    Calendar();
}
//实现月份向后翻页的方法
function addMonth()
{
    if((month+1)>11)
    {
        month=0;
        year=year+1;
    } else
    {
        month=month+1;
    }
    Calendar();
}
//判断用户自己输入的年份和月份
function setDate()
{
    if (document.all.month.value<1|document.all.month.value>12)
    {
        alert("月的有效范围在 1~12 之间!");
        return;
    }
    year=Math.ceil(document.all.year.value);
    month=Math.ceil(document.all.month.value-1);
    Calendar();
}
//设置按钮的样式
function buttonOver()
{
    var obj = window.event.srcElement;
    obj.runtimeStyle.cssText = "background-color:#FFFFFF";
}
function buttonOut()
{
    var obj = window.event.srcElement;
    window.setTimeout(function(){obj.runtimeStyle.cssText = "";},300);
}
</script>

```



本例涉及的样式表代码如下所示：

```
<Style>
  Input {font-family: verdana;font-size: 9pt;text-decoration: none;background- color:
#FFFFFF;height: 20px;border: 1px solid #666666;color:#000000;}
  .Calendar {font-family: verdana;text-decoration: none;width: 170;background- color:
#C0D0E8;font-size: 9pt;border:0px dotted #1C6FA5;}
  .CalendarTD {font-family: verdana;font-size: 7pt;color: #000000;background- color:
#f6f6f6;height: 20px;width:11%;text-align: center;}
  .Title {font-family: verdana;font-size: 11pt;font-weight: normal;height: 24px;text
-align: center;color: #333333;text-decoration: none;background-color: #A4B9D7;border-
top-width: 1px;border-right-width: 1px;border-bottom-width: 1px;border-left-width: 1px;
border-bottom-style:1px;border-top-color: #999999;border-right-color: #999999;border-
bottom-color: #999999;border- left-color: #999999;}
  .Day {font-family: verdana;font-size: 7pt;color:#243F65;background-color:
#E5E9F2;height: 20px;width:11%;text-align: center;}
  .DaySat {font-family: verdana;font-size: 7pt;color:#FF0000;text-decoration: none;
background-color:#E5E9F2;text-align: center;height: 18px;width: 12%;}
  .DaySun {font-family: verdana;font-size: 7pt;color: #FF0000;text-decoration: none;
background-color:#E5E9F2;text-align: center;height: 18px;width: 12%;}
  .DayNow {font-family: verdana;font-size: 7pt;font-weight: bold;color: #000000;
background-color: #FFFFFF;height: 20px;text-align: center;}
  .DayTitle {font-family: verdana;font-size: 9pt;color: #000000;background- color: #C0D0E8;
height: 20px;width:11%;text-align: center;}
  .DaySatTitle {font-family: verdana;font-size: 9pt;color:#FF0000;text-decoration: none;
background-color:#C0D0E8;text-align: center;height: 20px;width: 12%;}
  .DaySunTitle {font-family: verdana;font-size: 9pt;color: #FF0000;text-decoration: none;
background-color: #C0D0E8;text-align: center;height: 20px;width: 12%;}
  .DayButton {font-family: Webdings;font-size: 9pt;font-weight: bold;color: #243F65;
cursor:hand;text-decoration: none;}
</Style>
```

需要在 body 中添加一个表格，并在加载事件中调用“Calendar”方法，代码如下所示：

```
<body onload=" Calendar()">
<table border="0" cellpadding="0" cellspacing="1" class="Calendar" id="caltable">
<thead>
  <tr align="center" valign="middle">
    <td colspan="7" class="Title">
      <a href="javascript:subMonth();" title="上一月" Class="DayButton">3</a>
      <input name="year" type="text" size="4" maxlength="4" onkeydown="if (event.keyCode
==13){setDate()}" onkeyup="this.value=this.value.replace(/[\^0-9]/g, '')" onpaste="this.
value= this.value.replace(/[\^0-9]/g, '')"> 年 <input name="month" type="text" size="1"
maxlength="2" onkeydown="if (event.keyCode==13){setDate()}" onkeyup="this.value=this.
value.replace(/[\^0-9]/g, '')" onpaste="this.value=this.value.replace(/[\^0-9]/g, '')"> 月 <a
href="JavaScript: addMonth();" title="下一月" Class="DayButton">4</a>
    </td>
  </tr>
<tr align="center" valign="middle">
```

```

<script language="javascript">
    document.write("<td class=DaySunTitle id=diary >" + days[0] + "</td>");
    for (var intLoop = 1; intLoop < days.length-1;intLoop++)
        document.write("<td class=DayTitle id=diary>" + days[intLoop] + "</td>");
        document.write("<td class=DaySatTitle id=diary>" + days[intLoop] + "</td>");
</script>
</tr>
</thead>
<tbody border="1" cellspacing="0" cellpadding="0" ID="calendar" ALIGN="CENTER" ONCLICK=
"getDiary()">
<script language="javascript">
    for (var intWeeks = 0; intWeeks < 6; intWeeks++)
    {
        document.write("<TR style='cursor:hand'>");
        for (var intDays = 0; intDays < days.length;intDays++) document.write("<TD class=
CalendarTD onmouseover='buttonOver();' onmouseout='buttonOut();'></TD>");
        document.write("</TR>");
    }
</script>
</tbody>
</table>
</body>

```

【运行效果】

日历的运行效果如图 13-24 所示。

【难点剖析】

本例的难点是对天数的精确计算，尤其是闰年的二月份。本例中计算闰年使用的表达式是“`((0 == year % 4) && (0 != (year % 100))) || (0 == year % 400) ? 29 : 28;`”如果是闰年，则二月份的天数为 29，否则是 28。



图 13-24 日历的运行效果

13.32 生日提醒器

【实例描述】

同学录的相关网页一般会显示最近过生日的同学姓名，这显得比较人性化。本例学习如何制作这种生日提醒器。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
    function birthday(year,month,date,person) //一个生日对象，包括年、月、日和人名
    {
        this.year=year
        this.month=month

```



```

        this.date=date
        this.person=person
    }
    function personList()                                //创建一个空的列表对象
    { }
    pList=new personList()                                //创建用户列表项
    pList[0]= new birthday(82,1,16,"张三")                //在列表项中添加生日对象
    pList[1]= new birthday(33,9,27,"李四")
    pList[2]= new birthday(66,3,1,"王五")
    pList[3]= new birthday(79,4,27,"赵六")
    pList[4]= new birthday(80,6,2,"小明")
    pList[5]= new birthday(50,11,24,"晓红")
    var now=new Date()                                    //获取今天的日期
    today=new Date(now.getYear(),now.getMonth(),now.getDate())
                                                        //设置一个对象，一般为年月日 0:00:00

    function daysFromToday(sdate) {
    return Math.round((sdate.getTime()-today.getTime())/(24*60*60*1000))
                                                        //返回一个指定日期距今天的时间
    }
    function writeNextBirthday(list)
    {
        var daysToClosest=366
        var closest
        for (var i in list)                                //遍历列表中的每一项
        {
            thisDate=new Date(today.getYear(),list[i].month,list[i].date)
                                                        //获取朋友生日的月日
            if (daysFromToday(thisDate)<0)                //如果生日已经过了，则计算到下一年
                thisDate.setYear(today.getYear()+1)
            if (daysFromToday(thisDate)<daysToClosest) {
                daysToClosest=daysFromToday(thisDate)//计算距生日的天数-找出最近的一个
                closest=i
            }
        }
        if (daysToClosest==0)
            document.write("<B>今天 "+list[closest].person+" 已经 "+(today.getYear ()-list
[closest].year)+" 岁了 !!!</B><P>")
        else if (daysToClosest==1)
            document.write("明天 "+list[closest].person+" 即将 "+(today.getYear ()-list
[closest].year)+" 岁了!<P>")
        else
            document.write("最近一个过生日的是"+list[closest].person+" 还有 "+daysToClosest+"
天.<P>")
    }
</SCRIPT>
</head>
<body>
<SCRIPT LANGUAGE="JavaScript">

```

```

writeNextBirthday(pList)           //动态输出最近要过生日的对象
</SCRIPT>
</body>
</html>

```

【运行效果】

生日提醒器的效果如图 13-25 所示。

【难点剖析】

本例使用一个对象数组包含了所有朋友列表，然后根据当前日期与列表中的月和日进行比较，使用“daysToClosest”变量获取最近的一个朋友生日的日期。本例中注意日期的比较和设置。

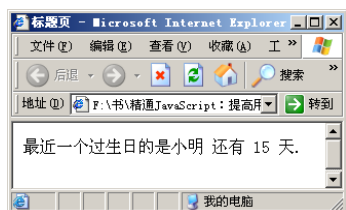


图 13-25 生日提醒器的效果

13.33 时间的倒影

【实例描述】

很多个性化的网站，为了显示页面的 3D 感觉，一般为页面添加倒影、水印等特效。本例学习如何显示当前时间的倒影。

【实现代码】

```

<script Language="JavaScript">
    var circletimer;
    function init()
    {
        if (document.all)
        {
            //初始化两个 div 的默认位置
            time2.style.left=time1.style.posLeft;
            time2.style.top=time1.style.posTop+time1.offsetHeight-5;
            settimes();
        }
    }
    function settimes()
    {
        //获取当前时间
        var time= new Date();
        hours= time.getHours();
        mins= time.getMinutes();
        secs= time.getSeconds();
        if (hours<10)
            hours="0"+hours;
        if(mins<10)
            mins="0"+mins;
        if (secs<10)
            secs="0"+secs;
    }

```



```
time1.innerHTML=hours+":"+mins+":"+secs;
time2.innerHTML=hours+":"+mins+":"+secs;
circletimer=setInterval('settimes()',1000); //循环时间显示
}
</script>
```

为了突出效果，本例使用样式表来定义时间的显示格式，代码如下所示：

```
<style type="text/css">
.time{font-family : Comic Sans Ms;
font-size : 16pt;font-weight : bold;color: #0F808F;}
</style>
```

在页面的 body 中，添加两个 div 用来显示两个时间，代码如下所示：

```
<div Id="time1" Style="position:absolute; width:8; height:8; top:9; left:35"
class="time"></div>
<div Id="time2" Style="position:absolute; filter:flipv() alpha(opacity=30);
font-style:italic"class="time"></div>
```

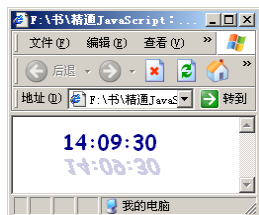


图 13-26 时间倒影的运行效果
“fliph”是水平翻转。

【运行效果】

时间倒影的运行效果如图 13-26 所示。

【难点剖析】

本例的重点是样式表 CSS+JavaScript 的应用，其中 CSS 定义倒影效果，JavaScript 的定时器“setInterval”循环显示时间。CSS 中的“filter:flipv()”是用来设置翻转效果的滤镜，其中“flipv”是垂直翻转，“fliph”是水平翻转。

13.34 使用正则表达式验证日期

【实例描述】

正则表达式是专门用来判断文本是否符合规范的表达式。本例使用正则表达式判断日期格式的正确性。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
//为 string 对象添加是否是正确日期属性
String.prototype.isDate = function()
{
    var r = this.match(/^(\\d{4})(-|\\/)(\\d{2})\\2(\\d{2})$/); //正则表达式
    if(r==null)return false;
    var d = new Date(r[1], r[3]-1, r[4]);
```

```

    return(d.getFullYear()==r[1]&&(d.getMonth()+1)==r[3]&&d.getDate() ==r[4]);
}
alert("2007-05-31".isDate());           // 正确的日期
alert("2006-1-41".isDate());           // 错误的日期
</script>
</head>
<body>
</body>
</html>

```

【难点剖析】

正则表达式使用“match”方法标记，返回“null”则表示指定的字符串不符合表达式的规范。详细的正则语法可参考相关资料。

13.35 全面的日期选择功能

【实例描述】

本例提供一些常用的日期换算，如本周、本月、上周等。可在代码中直接调用本例的方法实现日期的换算。

【实现代码】

```

<SCRIPT LANGUAGE="JavaScript">
    function today(){
        var now=new Date();           // 获取当前日期
        var tmpStr = now.getYear()+ "-"; // 当前年
        tmpStr += now.getMonth()+1 + "-"; // 当前月
        tmpStr += now.getDate();         // 当前日
        date1.value=tmpStr;
        date2.value=tmpStr;           // 显示当前的年、月、日连接字符串
    }

    function yesterday(){             // 获取昨天的日期
        var now=new Date();
        var tmpStr = now.getYear()+ "-"; // 当前年
        tmpStr += now.getMonth()+1 + "-"; // 当前月
        tmpStr += (now.getDate() - 1);    // 当前日
        date1.value=tmpStr;
        date2.value=tmpStr;           // 显示昨天的年、月、日连接字符串
    }

    function toYear(){
        var now=new Date();
        var tmpStr;
        tmpStr = now.getYear()+ "-";     // 当前年
        date1.value=tmpStr+"1-1";        // 年的开始
    }

```



```

        date2.value=tmpStr+"12-31";           //年的结束
    }

    function toMonth(){
        var now=new Date();
        var tmpStr,dt;
        tmpStr = now.getFullYear()+ "-";           //当前年
        tmpStr += now.getMonth()+1 + "-"; //月份+1
        tmpStr += (now.getDate() - now.getDate()+1); //天数+1
        date1.value=tmpStr;
        dt = new Date(now.getFullYear(), now.getMonth() + 1, 0);
        date2.value=replStr(dt.toLocaleDateString());
    }

    function lastWeek(){                         //上周
        var now=new Date();
        var currentWeek = now.getDay();           //获取一周中的第几天
        if ( currentWeek == 0 )                   //0 在英文中表示第一天, 中文表示最后一天
        {
            currentWeek = 7;
        }

        var monday = now.getTime() - (currentWeek+6)*24*60*60*1000;
        monday = replStr(new Date(monday).toLocaleDateString()); //获取周一
        var sunday = now.getFullYear()+ "-";
        sunday += now.getMonth()+1 + "-";           //获取周日
        if ( currentWeek == 7 )
        {
            sunday += (now.getDate() - 7);
        }else{
            sunday += (now.getDate() - currentWeek);
        }
        date1.value=monday;
        date2.value=sunday;                         //显示周一到周日的日期
    }

    function toWeek(){
        var now=new Date();
        var currentWeek = now.getDay();
        if ( currentWeek == 0 )
        {
            currentWeek = 7;
        }
        var monday = now.getTime() - (currentWeek-1)*24*60*60*1000; //获取周一
        var sunday = now.getTime() + (7-currentWeek)*24*60*60*1000;
        monday = replStr(new Date(monday).toLocaleDateString()); //获取周日
        sunday = replStr(new Date(sunday).toLocaleDateString());
        date1.value=monday;
    }

```



```

        date2.value=sunday;                                //显示周一到周日的日期
    }

    function replStr(str)
    {
        str = str.replace("年","-");                      //替换字符串——显示中文日期
        str = str.replace("月","-");
        str = str.replace("日","");
        return str;
    }
</SCRIPT>

```

【运行效果】

本例的运行效果如图 13-27 所示。

【难点剖析】

本例中的方法都是对日期对象进行换算，实现所要求的数据。其中“toYear”表示获取当前年的开始日期和结束日期，“toMonth”获取当前月的开始日期和结束日期，“lastWeek”获取上周的开始日期和结束日期。

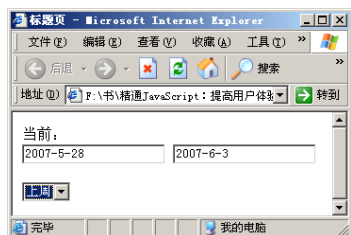


图 13-27 本例的运行效果

13.36 全球的时间查看表

【实例描述】

在宾馆的大堂经常看到全球各地的一些时间，这也可用在宾馆或航空部门的网页上，用户可通过选择时区来查看不同地方的时间。

【实现代码】

```

<script language="JavaScript">
var timerRunning = false;
var timezone = "格林尼治标准时间";
var adjust = 0;
function getTime(tzone, diff)                                //获取指定时区的时间
{
    if (timerRunning) {
        clearTimeout(updatetime);
        timerRunning = false;
    }
    gmtOffset=eval(diff+adjust);                              //此处设置时区差别
    timezone = tzone;
    checkDateTime();
}
function checkDateTime () {
var today = new Date();                                       //获取当前时间
var year = today.getYear() + 00;                             //获取年

```



```

var month = today.getMonth()+1; //获取月
var date = today.getDate(); //获取日期
var day = today.getDay(); //获取日
var hour = today.getHours(); //获取小时
var minute = today.getMinutes(); //获取分
var second = today.getSeconds(); //获取秒

var lastSat = date - (day+1);
while (lastSat < 32) lastSat+=7;
if (lastSat > 31) lastSat+=-7;
var firstSat = date - (day+1);
while (firstSat > 0) firstSat+=-7;
if (firstSat < 1) firstSat+=7;
if (((month == 4) && (date >= firstSat)) || month > 4) &&
(month < 11 || ((month == 10) && day <= lastSat))) adjust += 60;
yourOffset = (new Date()).getTimezoneOffset(); //当前计算机上的时间和 UTC 之间相差的分钟数
yourOffset = yourOffset + adjust;

if (((month == 4) && (date > 20)) || month > 4) && (month < 11 || ((month == 10) &&
day < 30))) adjust -= 60;

ourDifference = eval(gmtOffset - yourOffset); //根据本地时间和前面获取的与 utc 之间的差别
var half = eval(ourDifference % 60); //取除以 60 后的余数,剩下的是分钟数
ourDifference = Math.round(ourDifference / 60); //获取间隔的小时数
hour = eval(hour - ourDifference); //用本地小时——间隔的小时
var m = new Array("",
"1","2","3",
"4","5","6",
"7","8","9",
"10","11","12"); //月份数组
var leap = eval(year % 4); //判断闰年的变量(不太精确)

if ((half == -30) || (half == 30)) minute += 30;
if (minute > 59) minute -= 60, hour++; //当超过 60 分钟时,小时数增加
if (minute < 0) minute += 60, hour--; //当小于 60 分钟时,小时数减少
if (hour > 23) hour -= 24, date += 1; //当超过 24 小时时,天数加 1
if (((month == 4) || (month == 6) ||
(month == 9) || (month == 11)) && (date==31)) date = 1, month ++;
//指定的月为 30 天,超过 30,则月份加 1
if (((month == 2) && (date > 28)) && (leap != 0)) date = 1, month ++;
//二月份比较特殊
if ((month == 2) && (date > 29)) date = 1, month++; //非闰年时候的二月份
if (hour < 0) hour += 24, date --; //如果小时数小于 0,则天数减 1
if ((date == 32) && (month == 12)) month = m[1], date = 1, year++;
//当超过一年时
if (date == 32) date = 1, month++; //当超过一月时
if ((date < 1) && (month == 1)) month= m[12], date = 31, year--; //当前月份为一月份时,如果
天数小于 1,则转到 12 月份

```

```

if (date < 1) date = 31, month --; //日子小于1时,月份减1
if ((month == 4) || (month == 6) || //一月30天的设置
(month== 9) || (month == 11)) && (date == 31)) date = 30;
if ((month == 2) && (date > 28)) date = 29; //二月份的设置
if ((month == 2) && (date > 28)) && (leap != 0)) date=28;
for (i=1; i<13; i++) {
if (month == i) {
month = m[i];
break;
}
}

var dateTime = hour;
dateTime = ((dateTime < 10) ? "0:" : "") + dateTime; //显示两位数的时间
dateTime = " " + dateTime;
dateTime += ((minute < 10) ? ":0" : ":") + minute; //显示两位数的分钟
dateTime += ((second < 10) ? ":0" : ":") + second; //显示两位数的秒
dateTime += (hour >= 12) ? "下午, " : "上午, "; //显示汉字:上午和下午
dateTime += year + "年" + month + "月" + date + "日" ; //显示年、月、日

document.clock.zonetime.value = dateTime; //显示所选时区的时间
document.clock.zonename.value = timezone; //显示选择的时区
updatetime=setTimeout("checkDateTime()", 1000); //定时更新时间——每隔1秒
timerRunning = true;
}
</script>

```

【运行效果】

本例的运行结果如图 13-28 所示。

【难点剖析】

本例的重点有两个：格林尼治时间的定义，以及当地时间与标准时间之间的时间差。全球被划分为 24 个时区，以通过英国格林尼治天文台的本初子午线为标准，其东西经度 7.5 度的范围为零时区，每个时区中央经线上的时间就是各时区的标准时间。“getTimezoneOffset”方法用来获取当地时间与 UTC（标准时间）之间的时间差。因为此方法返回的是分钟数，所以可以通过“/60”的方式获取小时数，然后通过“%60”的方式获取剩余的分钟数。

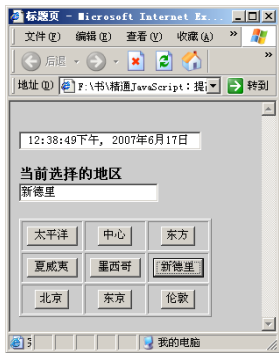


图 13-28 本例的运行结果

13.37 无刷新定时取数据

【实例描述】

客户端在获取服务器数据后，需要刷新页面再重新显示获取数据后的结果。考虑页面刷新的时间和效果问题，本例学习一种新方法，使用 Ajax 技术实现无刷新定时取数据。



【实现代码】

```

<HTML>
<HEAD>
<TITLE>天气预报</TITLE>
<script language="javascript">
    var xmlhttp;
    function getWeather()
    {
        //创建异步对象
        xmlhttp=new ActiveXObject("Msxml2.XMLHTTP");
        //加载服务器-注意 URL 参数的使用
        xmlhttp.Open("GET","http://tw.weather.yahoo.com/world_single.html?city=
10101",false)
        //异步对象与事件挂钩
        xmlhttp.onreadystatechange=stateChange;
        //发送请求——无参数
        xmlhttp.Send(null);
    }
    function stateChange()
    {
        if(xmlhttp.readyState==4 && xmlhttp.status==200)
        {
            //获取所有返回的数据
            var data=xmlhttp.responseText;
            //过滤自己需要的数据
            var begin=data.indexOf("国际个别都市 start");
            var end=data.indexOf("国际个别都市 end");
            var weather=data.substring(begin+15,end);
            //填充天气内容
            document.getElementById("divweather").innerHTML=weather;
            //显示结果
            document.getElementById("divweather").style.visibility="visible";
        }
    }
    window.setInterval("getWeather()", 3000);
</script>
</HEAD>
<BODY onload="getWeather()">
<div align="center" id="today_time">当天的日期
</div>
<div align="center" id="divweather"></div>
<script language="javascript">
//设置显示星期几，用数组存储
var x = new Array("星期日", "星期一", "星期二");
var x = x.concat("星期三", "星期四", "星期五");
var x = x.concat("星期六");
var today_time = new Date();
//获取当天的日期

```

```
//先后用中文表示的日期
document.all("today_time").innerText=today_time.getFullYear()+'年'+(today_ time.
getMonth()+1)+'月'+today_time.getDate()+'日\n'+x[today_time.getDay()];

</script>
</BODY>
</HTML>
```

【运行效果】

本例的运行效果如图 13-29 所示。每隔三秒会更新数据，看看页面是否有刷新。



图 13-29 本例的运行效果

【难点剖析】

本例使用 XMLHttpRequest 对象从其他网络获取天气预报信息,此处重点是 XMLHttpRequest 的使用步骤。使用“window.setInterval”定时器，每隔三秒从服务器重新获取一次数据，这样能始终保持最新的天气预报信息。

13.38 取当月的最后一天

【实例描述】

在设计财务软件时，经常会遇到计算当月最后一天的情况。本例使用 JavaScript 计算当月的最后一天。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function getLastDay(year,month)
{
    var new_year = year; //取当前的年份
    var new_month = month++; //取下一个月的第一天，方便计算（最后一天不固定）
```



```
if(month>12) //如果当前大于12月，则年份转到下一年
{
    new_month -=12; //月份减
    new_year++; //年份增
}
var new_date = new Date(new_year,new_month,1); //取当年当月中的第一天
return (new Date(new_date.getTime()-1000*60*60*24)).getDate(); //获取当月最后一天日期
}
</script>
<body>
    <input id="Button1" type="button" value="取 2007 年 5 月的最后一天" onclick="alert
(getLastDay(2007,5))" />
</body>
</html>
```

【难点剖析】

本例的重点是如何获取最后一天。因为月份的最后一天是不固定的，有时 31 天，也有时 30 天，但月份的第一天是固定的，所以先找到下一月的第一天，然后用此日期减一则是指定月的最后一天。

第 14 章

数字、数组和统计函数特效

本章导读

函数就是 JavaScript 中常说的方法，本章提供几个常用函数，读者可以直接将其添加到网站项目中。本章还介绍一些常用的计算方法和技巧，在实际项目中的应用非常广泛。



14.1 边打字边显示字数

【实例描述】

有一些文本框限定了输入字数，为了帮助用户正确操作，会有字数提示功能。本例学习如何实现这种字数提示功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function Check(txt)
{
    TextCount=txt.value.length;           //获取文本框的长度
    div1.innerText=TextCount;             //将长度显示在div中
}
</script>
</head>
<body>
<textarea type=text id=txt1 onkeyup="Check(this) " onKeyDown="Check(this)">
</textarea><br />当前字符数: <div id="div1"></div>
</body>
</html>
```

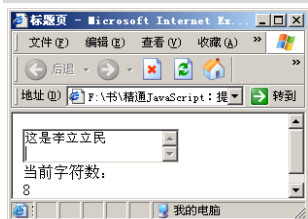


图 14-1 字数提示效果

【运行效果】

字数提示效果如图 14-1 所示。

【难点剖析】

本例的重点是文本框事件的应用。“onkeyup”是按键弹起时的触发事件，“onKeyDown”是按键按下时的触发事件。“Check”方法用来监测文本的长度，并在 div 中动态地显示当前的字数。

14.2 创建随机数

【实例描述】

几乎所有的开发语言都有一个用来生成随机数的方法。本例学习如何使用 JavaScript 创建随机数。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
```



```

</head>
<body>
<script LANGUAGE="JavaScript">
    rnd.today=new Date();           //获取当前时间
    rnd.seed=rnd.today.getTime();   //获取一个毫秒数
    function rnd() {                 //返回浮点数类型的随机数
        rnd.seed = (rnd.seed*9301+49297) % 233280;
        return rnd.seed/(233280.0); //返回随机浮点数
    };
    function rand(number) {          //返回整数类型的随机数（1~10 之间）
        return Math.ceil(rnd()*number); //返回随机整数
    };
</script>
<input type=button name="btn1" value="获取整数随机数" onClick="alert(rand(20))">
<input type=button value="获取浮点数随机数" onClick="alert(rnd())">
</body>
</html>

```

【难点剖析】

本例的重点是如何创建一个随机数。本例使用“getTime”方法获取一个整数，这个整数代表了从 1970 年 1 月 1 日，到当前日期之间的毫秒数。然后使用此毫秒数再创建一个随机种子，以生成随机浮点数。随机整数的生成依靠随机浮点数。

14.3 JavaScript 创建二维数组

【实例描述】

JavaScript 提供一个数组对象 Array，但其默认只是一维数组。本例通过一个级联菜单学习如何创建二维数组。

【实现代码】

```

<HTML>
<HEAD>
<TITLE>二维数组示例</TITLE>
</HEAD>
<SCRIPT LANGUAGE=javascript>
//定义二维数组 oArray,用于存放城市名称
var aCity=new Array();
aCity[0]=new Array();
aCity[1]=new Array();
aCity[2]=new Array();
aCity[3]=new Array();
//为二维数组赋值
aCity[0][0]="--请选择--";
aCity[1][0]="--请选择--";
aCity[1][1]="朝阳区";

```



```

aCity[1][2]="海淀区";
aCity[1][3]="东城区";
aCity[1][4]="西城区";
aCity[2][0]="--请选择--";
aCity[2][1]="济南市";
aCity[2][2]="青岛市";
aCity[2][3]="潍坊市";
aCity[3][0]="--请选择--";
aCity[3][1]="浦东区";
aCity[3][2]="徐汇区";
aCity[3][3]="虹桥";
function selectCity()
{
var i,iIndex;
iIndex=document.form1.oProvince.selectedIndex;           //获取选择的省或区
iCityCount=0;
while (aCity[iIndex][iCityCount]!=null)
    iCityCount++;                                           //计算选定省份的市或区个数
document.form1.oCity.length=iCityCount;                   //改变第二个下拉框的选项数
for (i=0;i<=iCityCount-1;i++)
    document.form1.oCity[i]=new Option(aCity[iIndex][i]);  //创建区或市的下拉列表
document.form1.oCity.focus();                               //第二个下拉框获得焦点
}
</SCRIPT>
<BODY onfocus=selectCity(>
<H3>选择所在的省份及城市</H3>
<FORM NAME="form1">
<P>省份:
<SELECT NAME="oProvince" SIZE="1" ONCHANGE=selectCity(>
    <OPTION>--请选择--</OPTION>
    <OPTION>北京</OPTION>
    <OPTION>山东省</OPTION>
    <OPTION>上海</OPTION>
</SELECT>
</P>
<P>城市:
<SELECT NAME="oCity" SIZE="1">
    <OPTION>--请选择--</OPTION>
</SELECT>
</P>
</FORM>
</BODY>
</HTML>

```

【运行效果】

二维数组实现的下拉列表效果如图 14-2 所示。

【难点剖析】

本例的重点是二维数组的创建。首先使用 Array 创建一个一维数组“aCity”，然后将此数组

中的每个项又设置为一维数组。这样就构建了一个简单的二维数组。注意每个数组的赋值方式。

14.4 截断小数点位数

【实例描述】

当用户不知道数据库中数据的小数位数时,可通过程序判断用户输入的数据,自动截取到需要的小数位数。本例演示如何自动截断小数点的位数。

【实现代码】

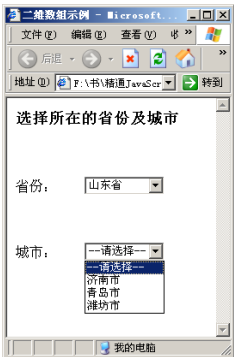


图 14-2 二维数组实现的下拉列表效果

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
function sts()
{
    var txt=eval(document.getElementById("Text1").value);    //获取要截取的数据
    alert(txt.toFixed(2));    //显示截取后的数据——结果是 12345.27
}
</script>
</head>
<body>
    <input id="Text1" type="text" value="12345.267514" />
    <input id="Button1" type="button" value="截取 2 位" onclick="sts()" />
</body>
</html>
```

【难点剖析】

本例的重点是“eval”函数和“toFixed”方法。“eval”函数可以将字符型数据转换为数值型。“toFixed”方法返回一个字符串类型的数字,该字符串中小数点之前必须至少有一位有效数字,而且其后必须包含参数中指定位数的数字。

14.5 删除数组中指定元素

【实例描述】

数组对象以有序的方式存储类型相同的数据,可通过索引查找数据元素,但数组对象不提供删除指定元素的功能。本例将创建一个删除方法,用来实现删除指定元素的功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
```



```
<script type=text/javascript>
// 自定义数组删除方法
Array.prototype.del = function(n)
{
  if (n<0) return this;
  return this.slice(0,n).concat(this.slice(n+1,this.length));
}
var arr = new Array("1","2","3","4");           //定义 4 个元素的数组
alert(arr.del(1))                               //删除数组中第二个元素，从 0 开始索引
//-->
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的难点是“prototype”。“prototype”提供对象的一组基本功能，可以为 JavaScript 中的已有对象添加一些属性，本例中为 Array 对象添加了“del”方法。“slice”方法复制指定位置的元素，本例用来除去要删除的指定元素。

14.6 数字选中后放大

【实例描述】

在 Web 页面中分页时，为了突出显示用户选择的页数，通常会放大选中页的页码。本例学习如何放大被选中的数字。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
  <style type="text/css">
    /* 定义 ul 根级目录项的样式*/
    ul#menu{
      list-style-type: none;
      margin: 50px;
      width:200px;
      float: left;
      display: inline;
      clear: both;
    }
    /* 定义 ul 根级目录下 li 项的样式*/
    ul#menu li{
      float: left;
      display: inline;
```

```

width:20px;
height: 20px;
margin: 2px;
}
/* 定义 ul 根级目录下 li 项中链接的样式*/
ul#menu li a {
text-decoration: none;
display: block;
width:20px;
height:20px;
border:1px red solid;
background-color: White;
line-height: 20px;
font-size: 12px;
text-align: center;
}
/* 定义 ul 根级目录下 li 项中鼠标滑过链接时的样式*/
ul#menu li a:hover{
position: absolute;
width:40px;
height: 40px;
line-height: 40px;
font-size: 32px;
z-index:100;
margin: -10px 0 0 -10px;
}
/* 定义 ul 根级目录下鼠标滑过 li 时的样式*/
ul#menu li:hover + li a{
position: absolute;
width:30px;
height: 30px;
line-height: 30px;
font-size: 24px;
z-index:99;
margin: -5px 0 0 -5px;
}
</style>
</head>
<body>
<ul id="menu">
<li><a href="#" title="big"><span>1</span></a></li>
<li><a href="#" title="big"><span>2</span></a></li>
<li><a href="#" title="big"><span>3</span></a></li>
<li><a href="#" title="big"><span>4</span></a></li>
<li><a href="#" title="big"><span>5</span></a></li>
<li><a href="#" title="big"><span>6</span></a></li>
<li><a href="#" title="big"><span>7</span></a></li>
</ul>

```

```
</body>
</html>
```

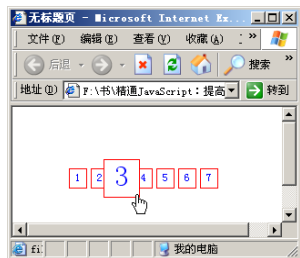


图 14-3 数字选中放大的效果

【运行效果】

数字选中放大的效果如图 14-3 所示。

【难点剖析】

本例主要通过 CSS 样式表实现数字项的放大和缩小。其中“ul#menu li a:hover”样式定义鼠标滑过时数字的样式，通过更改当前数字的大小和 4 个边距，最终实现选中数字的放大效果。

14.7 统计字符数的方法

【实例描述】

用户在文本框内输入英文字符时，可以根据用户输入计算每个字符的输入数量。本例就学习这种计算方法。

【实现代码】

```
<html>
<head>
<title>输出测试</title>
<script language="javascript">
    function cal()
    {
        var calTxtArr = []; //创建数组
        for (var i = 0, s = document.all.myContent.value; i < s.length; ++i) //遍历文本中所有元素
        {
            var c = s.charAt(i); //获取指定元素
            calTxtArr[c] = calTxtArr[c] == null ? 1 : calTxtArr[c] + 1; //判断是否已经开始计算
        }
        var bufferArr = [];
        for (var c in calTxtArr) //如果已经存在
        {
            bufferArr.push(c + ": " + calTxtArr[c]); //添加到数组中
        }
        document.all.calTxt.value = bufferArr.join("\n"); //显示在文本框内，并换行
    }
</script>
</head>
<body>
输入内容: <br>
<textarea rows="5" cols="40" name="myContent"></textarea>
<input type="button" value="开始计算" onclick="cal()">
```

```
<br><br>
<textarea name = "calTxt" rows = "20" cols = "40" readonly>
</textarea>
</body>
</html>
```

【运行效果】

本例的运行效果如图 14-4 所示。

【难点剖析】

本例的重点是使用一个缓冲数组保存已经开始计算的字符。首先遍历用户输入的字符，根据字符在缓存数组中是否存在判断字符的数量。

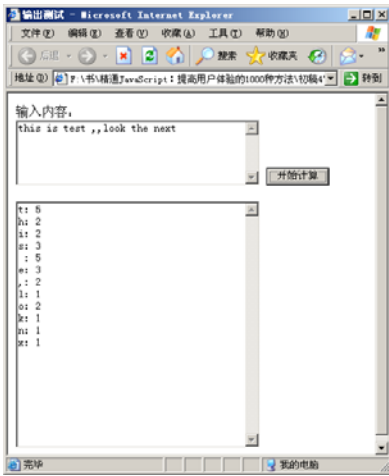


图 14-4 本例的运行效果

14.8 JavaScript 遍历数组

【实例描述】

数组是常用的数据保存方式，可用来保存相同类型的多组数据。本例学习如何遍历读取数组中的每一个数据。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
var a=new Array("a","b","c");           //创建数组
for(i in a)                             //遍历数组中的每项
{
    alert(a[i]);                         //显示每一个元素
}
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是数组的创建和遍历。在 JavaScript 中数组是一个对象，用“Array”表示。创建数组使用“new Array”实现。遍历数组使用“for…in”语句。

14.9 获取字符串型数组索引的数组长度

【实例描述】

创建数组时默认的索引是数值型，但有时候因为类型的需要会将索引设置为字符型。本例



学习如何遍历字符型索引的数组。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
var info = new Array();           //创建数组对象
info['name'] = '张三';           //设置数组中的一项，注意索引为字符串类型
info['age'] = '26';               //设置数组中的第二项
var i=0;
for(var n in info)                //遍历数组中的每一项
{
    i++;                           //用来获取数组中的元素个数
}
alert("总共"+i+"个元素");         //显示数组中总共多少元素
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的难点是如何遍历一个数组。在 JavaScript 中使用“for...in”语句，注意“var n”表示的是每次遍历的当前元素，“in”后面的是被遍历的数组。

14.10 用 JavaScript 实现数组排序

【实例描述】

数组排序有多种方法，本例将使用最简单的插入式排序法，学习如何实现数组排序。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
//为参数中的数组排序
function Sort() {
    var str=document.getElementById("txt1").value;
    var arr=str.split(',');
    var st = new Date();                //计算排序的开始时间
    var temp, j;
    for(var i=1; i<arr.length; i++) {   //逐个检查数组中的元素
        if((arr[i]) < (arr[i-1])) {
            temp = arr[i];
```



```

    j = i-1;
    do {
        arr[j+1] = arr[j];           //数据的前后替换
        j--;
    }
    while (j>-1 && (temp) < (arr[j]));
    arr[j+1] = temp;
}
}
status = (new Date() - st) + ' ms'; //用结束时间减去开始时间
alert(arr);                         //返回排序后的结果
}
</script>
</head>
<body>
<input type=text name="txt1" value="2,8,4,1,3,6">
<input type=button value="排序" onClick="Sort()">
</body>
</html>

```

【运行效果】

排序后的结果如图 14-5 所示。

【难点剖析】

本例的难点有两个：将输入框的内容转换为数组、对数组的数据进行排序。将一段字符串转换为数组使用“split”方法，其可以根据统一的间隔符号将字符转换。对数组排序使用的是逐个比较法，详细方法可参考代码中的“for”循环语句。

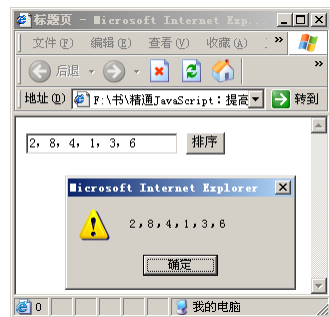


图 14-5 排序后的结果

14.11 数字千分位函数

【实例描述】

在财务运算中显示数字时，有时要求必须显示千分位符号。本例制作一个函数，使用正则实现任意小数和整数的千分位表现形式。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="JavaScript">
function millon()
{
    var s=parseFloat(myNum.value); //获取小数型数据

```

```
s+=" ";  
if(s.indexOf(".")== -1) s+="0"; //如果没有小数点，在后面补个小数点和 0  
if(/\.\d$/.test(s)) s+="0"; //正则判断  
while(/\d{4}(\.|,)/.test(s)) //符合条件则进行替换  
    s=s.replace(/(\d)(\d{3}(\.|,))/,"$1,$2"); //每隔三位添加一个，  
alert(s);  
}  
</script>  
<input type="text" name="myNum" value="1234567890.08"><input type="button" value=  
"million" onclick="million()">  
</body>  
</html>
```

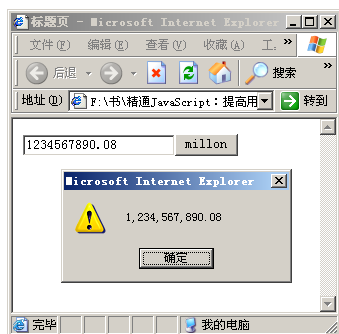


图 14-6 本例的运行效果

【运行效果】

本例的运行效果如图 14-6 所示。

【难点剖析】

本例使用正则进行循环，循环条件是判断“,”符号前是否超过三位数。如果是则使用“replace”方法，实现每隔三位输出千分位符号的特效。

14.12 读写 Cookie 的函数

【实例描述】

Cookie 是客户端用来保存数据的对象，本例将所有有关 Cookie 的操作封装成一个函数库。读者通过学习本例可以全面地掌握 Cookie 的使用。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
<script Language="JavaScript">  
//获得 Cookie 解码后的值  
function GetCookieVal(offset)  
{  
    var endstr = document.cookie.indexOf (";", offset);  
    if (endstr == -1)  
        endstr = document.cookie.length;  
    return unescape(document.cookie.substring(offset, endstr));  
}  
//设定 Cookie 值——将值保存在 Cookie 中  
function SetCookie(name, value)  
{  
    var expdate = new Date(); //获取当前日期  
    var argv = SetCookie.arguments; //获取 Cookie 的参数  
    var argc = SetCookie.arguments.length; //Cookie 的长度
```

```

var expires = (argc > 2) ? argv[2] : null;           //Cookie 有效期
var path = (argc > 3) ? argv[3] : null;             //Cookie 路径
var domain = (argc > 4) ? argv[4] : null;           //Cookie 所在的应用程序域
var secure = (argc > 5) ? argv[5] : false;          //Cookie 的加密安全设置
if(expires!=null) expdate.setTime(expdate.getTime() + ( expires * 1000 ));
document.cookie = name + "=" + escape (value) +((expires == null) ? "" : ("; expires="+
expdate.toGMTString()))
+((path == null) ? "" : ("; path=" + path)) +((domain == null) ? "" : ("; domain=" + domain))
+((secure == true) ? "; secure" : "");
}
//删除指定的 Cookie
function DelCookie(name)
{
var exp = new Date();
exp.setTime (exp.getTime() - 1);
var cval = GetCookie (name);                       //获取当前 Cookie 的值
document.cookie = name + "=" + cval + "; expires="+ exp.toGMTString();
                                                    //将日期设置为过期时间
}
//获得 Cookie 的值-name 用来搜索 Cookie 的名字
function GetCookie(name)
{
var arg = name + "=";
var argLen= arg.length;                             //指定 Cookie 名的长度
var cookieLen= document.cookie.length;              //获取 Cookie 的长度
var i = 0;
while (i < cookieLen)
{
var j = i + argLen;
if (document.cookie.substring(i, j) == arg)         //依次查找对应 Cookie 名的值
return GetCookieVal (j);
i = document.cookie.indexOf(" ", i) + 1;
if (i == 0) break;
}
return null;
}
</script>
</head>
<body>
<input type=text name="txt1" value="搜索引擎是百度">
<input type=button value="保存 Cookie" onClick="javascript:SetCookie('sousuo', txt1. value)">
<input type=button value="获取 Cookie" onClick=" javascript:alert(GetCookie ('sousuo'))">
</body>
</html>

```

【难点剖析】

本例的重点是“document.cookie”，在 JavaScript 中不管是保存 Cookie 还是获取 Cookie，都是使用“document.cookie”。具体的使用方法参考代码中的注释。



14.13 获取 JavaScript 函数中的所有参数

【实例描述】

JavaScript 的函数可以自定义多个参数。本例学习如何获取函数的所有参数，并显示这些参数的值。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<SCRIPT LANGUAGE="JavaScript">
    var Arr = [1,2,3]
    var Val = 10;
    function myFun(arg1,arg2,arg3)
    {
        var Args = myFun.arguments;
        for (var i=0;i<Args.length;i++)                //遍历所有的参数
        {
            alert(Args[i])                            //显示所有的参数
        }
    }
    myFun('myfile1',Arr,Val)
</SCRIPT>
</body>
</html>
```

【难点剖析】

本例的重点是函数的“arguments”属性，其用来获取函数的所有参数。“for”语句遍历所有的参数，并使用“Args[i]”显示每个参数的值。

14.14 奇偶数的判断

【实例描述】

奇数和偶数的判断是数学运算中经常碰到的问题。本例使用 JavaScript 来实现奇偶数的判断。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
```

```
<Script language="JavaScript" type="text/javascript">
function chk(num){
    alert( (num%2 ==0) ? "偶数":"奇数");           //判断是否能被 2 整除
}
</Script>
</head>
<body>
<input type="text" name="txt1" value="4">
<input type="button" value="判断" onClick="chk(txt1.value)">
</body>
</html>
```

【难点剖析】

本例的重点是一个三元运算符。三元运算符的使用语法如下所示：

表达式?值 1:值 2

其中表达式运算的结果必须是 bool（布尔）型，如果返回“true”则取“?”后面的第一个值，否则取第二个值。两个值之间用“:”间隔。

14.15 在 JavaScript 运行 VBScript 函数

【实例描述】

客户端脚本可以通过 JavaScript 和 VBScript 实现，两种语言提供的函数库不同。本例学习如何在 JavaScript 中调用 VBScript 函数。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="javascript">
execScript("MsgBox '欢迎使用'", "vbscript");
</script>
</body>
</html>
```

【运行效果】

本例的运行效果如图 14-7 所示。

【难点剖析】

本例的重点是“execScript”方法，其用来执行一段脚本。“alert”是 JavaScript 中弹出对话框的方法，而“MsgBox”是 VBScript 中弹出对话框的方法，本例使用“execScript”方法调用“MsgBox”方法实现了弹出式窗口的效果。

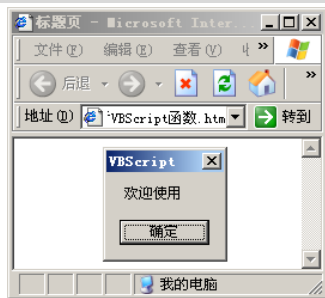


图 14-7 本例的运行效果



14.16 购物篮中常用的计算总价效果

【实例描述】

在一些购物网站中，用户需要先将商品放到虚拟的购物篮中，付款时系统会自动计算购物篮中商品的总价值。本例学习制作一个简单的购物篮。

【实现代码】

```
<html>
<head>
<title>小小购物篮</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<script language="JavaScript">
var messages = new Array(11); //价格数组
messages[0] = "";
messages[1] = "12.30 元";
messages[2] = "13.00 元";
messages[3] = "14.20 元";
messages[4] = "15.00 元";
messages[5] = "21.30 元";
messages[6] = "22.70 元";
messages[7] = "23.20 元";
messages[8] = "24.40 元";
messages[9] = "31.80 元";
messages[10] = "32.60 元";
function numChange()
{
    with (document.form1)
    {
        var numIndex = bookNum.selectedIndex; //判断用户选择的本数
        bookPrice.value = messages[numIndex]; //获取价格
        bookTotal.value = messages[numIndex].slice(0,-1)*numIndex+"元"; //计算总价
    }
}
</script>
<title>无标题文档</title>
</head>

<body>
<form name="form1" method="post" action="">
数量 <select name="bookNum" onChange="numChange()">
    <option value="0">购买数量</option>
    <option>1 本</option>
```

```
<option>2 本</option>
<option>3 本</option>
<option>4 本</option>
<option>5 本</option>
<option>6 本</option>
<option>7 本</option>
<option>8 本</option>
<option>9 本</option>
<option>10 本</option>
</select>
单价
<input name="bookPrice"  readonly />
总价
<input name="bookTotal" type="text" id="bookTotal" />
</form>
</body>
</html>
```

【运行效果】

购物篮运行的效果如图 14-8 所示。

【难点剖析】

本例中的“单价”输入框是只读的，在 input 标签中使用“readonly”控制文本框是否只读。用户选择下拉框，系统根据用户的选择自动计算本商品的单价，这些单价都保存在数组“messages”中。计算总价时使用数组对象的“slice”方法，此方法也返回一个数组，该数组是原数组的子集，始于参数一，结束于参数二。

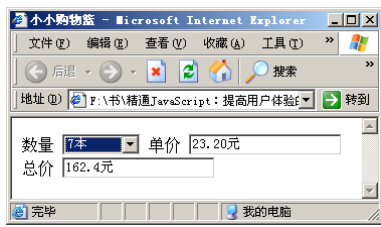


图 14-8 购物篮运行的效果

14.17 同一用户的来访统计

【实例描述】

很多网站将用户的登录信息保存，这样可以防止黑客的袭击，也可以方便用户的其他操作（如个性设置）。本例学习如何使用 Cookie 保存用户的访问信息。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
<script language="JavaScript">
  var caution = false;
  //将用户的访问信息保存在 Cookie 中
  function setCookie(name, value, expires, path, domain, secure)
  {
```



```
var curCookie = name + "=" + escape(value) +
((expires) ? "; expires=" + expires.toGMTString() : "") +
((path) ? "; path=" + path : "") +
((domain) ? "; domain=" + domain : "") +
((secure) ? "; secure" : "")
if (!caution || (name + "=" + escape(value)).length <= 4000)
    document.cookie = curCookie
else
if (confirm("COOKIE 超过 4KB 的部分将会丢失!"))
    document.cookie = curCookie
}
//根据 Cookie 名获取保存在 Cookie 中的数据
function getCookie(name)
{
var prefix = name + "="
var cookieStartIndex = document.cookie.indexOf(prefix)
if (cookieStartIndex == -1)
    return null
var cookieEndIndex = document.cookie.indexOf(";", cookieStartIndex + prefix.length)
if (cookieEndIndex == -1)
    cookieEndIndex = document.cookie.length
return unescape(document.cookie.substring(cookieStartIndex + prefix.length,
cookieEndIndex))
}
//删除 Cookie 信息
function deleteCookie(name, path, domain)
{
if (getCookie(name)) {
document.cookie = name + "=" +
((path) ? "; path=" + path : "") +
((domain) ? "; domain=" + domain : "") +
"; expires=Thu, 01-Jan-70 00:00:01 GMT" }
}
//计算并显示访问次数
function fixDate(date)
{
var base = new Date(0);
var skew = base.getTime();
if (skew > 0)
date.setTime(date.getTime() - skew)
}
var now = new Date();
fixDate(now);
now.setTime(now.getTime() + 365 * 24 * 60 * 60 * 1000);
var accesses= getCookie("counter");
if (!accesses)
    accesses=1 ;
else
```



```

        accesses= parseInt(accesses) + 1;
        setCookie("counter", accesses, now) ;
        document.write("您好,谢谢您的第" + accesses+ "次光临!") ;
    </script>
</head>
<body>
</body>
</html>

```

【运行效果】

用户访问网站时的提示信息如图 14-9 所示。可按 F5 键刷新页面测试,或者关闭网页再打开测试,可以发现统计数据保持增加的趋势。

【难点剖析】

本例的重点是 Cookie 的保存和读取。保存 Cookie 一般用“document.cookie=数据”的形式;读取 Cookie 则需要在 Cookie 中根据保存的 Cookie 名字进行搜索,本例中使用“document.cookie.indexOf”方法实现 Cookie 的查找。

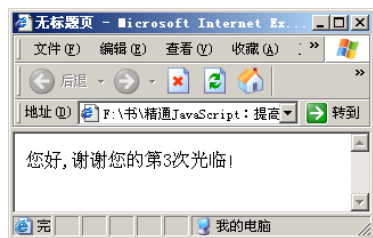


图 14-9 用户访问网站时的提示信息

14.18 16 进制数转换为 10 进制数

【实例描述】

计算机中数值的表示方法有很多,有时可以使用 16 进制表示法为数据加密。本例学习如何将 16 进制数据转换为 10 进制数据。

【实现代码】

```

<head>
<title>标题页</title>
<script language="javascript">
function change(sixteen) {
    var ten = parseInt(sixteen,16);           //使用字符串转换为整数的方法实现进制转换
    alert(ten);                               //显示转换后的结果
}
</script>
</head>
<body>
<span>
    <input id="Text1" type="text" /><input id="Button1" type="button" value="转换" onclick=
"change(Text1.value)" />
</body>
</html>

```

【难点剖析】

本例的重点是“parseInt”方法,此方法用来将字符串类型转换为数值类型。其使用语法为



【实例描述】

在网络中传递 URL 时，为了保证传递的安全，有时需要将 URL 加密。本例使用 16 进制表示传送的 URL。

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
function ConvertUrl(str)
{
    var newStr = '';
    for(var i=0; i < str.length; i++)
    {
        var ch = str.charAt(i);
        var code = str.charCodeAt(i);
        newStr += code.toString(16) ;
    }
    alert( newStr);
}
</script>
</head>
<body>
    <input id="Button1" type="button" value="转化" onclick="ConvertUrl ('http://WWW.
LE.COM')" />
</body>
</html>
```

16 进制的 URL 输出效果如图 14-10 所示。

本例的重点是从字符串到 16 进制代码的转换。首先逐个获取 URL 中的字符，然后得到字符的键值，最后使用“toString(16)”方法获取字符的 16 进制代码。

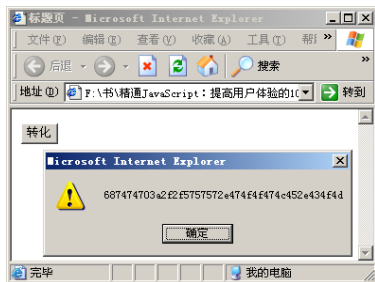


图 14-10 16 进制的 URL 输出效果

【实例描述】

大写金额是我国特有的一种金额表现形式。本例学习如何将阿拉伯数字形式的金额转换为

大写金额。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="JavaScript">
function daxie()
{
    //定义大写数组
    this.values = ["零", "壹", "贰", "叁", "肆", "伍", "陆", "柒", "捌", "玖"];
    this.digits = ["", "拾", "佰", "仟"];
}

function daxie.prototype.getDaXie(money)
{
    if(isNaN(money)) return ""; //如果不是数值型，直接返回空
    var number = Math.round(money*100)/100; //取数值的整数
    number = number.toString(10).split('.'); //整数和小数分开
    var moneyInt = number[0]; //整数部分
    var len = moneyInt.length; //整数的长度
    if (len > 12) //长度如果超出范围
        return "数值超出范围! 支持的最大数为 999999999999.99!";
    var returnValue = this.millonTrans(moneyInt.slice(-4));
    if (len > 4) //多于万位
        returnValue = this.millonTrans(moneyInt.slice(-8,-4)) + (moneyInt.slice (-8,-4)!
="0000"? "万":"" ) + returnValue;
    if (len > 8) //多于亿位
        returnValue = this.millonTrans(moneyInt.slice(-12,-8)) + "亿" + returnValue;
    if(returnValue!="")
        returnValue += "圆"; //添加最后一个字符
    if(number.length==2) //是否是带小数的金额
    {
        var cok = number[1].split('');
        if(returnValue!=" " || cok[0]!="0")
            returnValue += this.values[parseInt(cok[0])] + (cok[0]!="0"? "角":"" );
        //十位数显示角

        if(cok.length>=2)
            returnValue += this.values[parseInt(cok[1])] + "分"; //个位数显示分
    }
    if(returnValue!=" " && !/分$/.test(returnValue)) //使用正则判断是否有小数
        returnValue += "整";
    return returnValue;
}

function daxie.prototype.millonTrans(strTemp)
{
    var money = strTemp.split(''); //将金额转换为数组

```



```
var mLength = money.length-1; //金额的长度
var returnValue = "";
for (var i=0; i<=mLength; i++) //遍历每个元素
    returnValue += this.values[parseInt(money[i])] + (money[i]!='0'?this.digits
[mLength-i]: "");
    returnValue = returnValue.replace(/零+$/, "").replace(/零{2,}/, "零");
    //返回转换后的数值

return returnValue;
}

var stmp = "";
var daXieM = new daXie();
function strTrans(strT)
{
    if(strT.value==stmp) return;
    var ms = strT.value.replace(/[^\d\.]/g, "").replace(/(\.\d{2})+$/, "$1");
    //验证用户的输入

    var txt = ms.split("."); //分割成数组
    while(/\\d{4}(,|\\$)/.test(txt[0]))
        txt[0] = txt[0].replace(/\\d{4}(,|\\$)/, "$1,$2"); //科学计数法表示形式
    strT.value = stmp = txt[0]+(txt.length>1?"."+txt[1]: "");
    daXieTxt.value = daXieM.getDaXie(parseFloat(ms)); //显示大写
}
</script>
小写金额: <input type="text" name="xiaoxieTxt" onkeyup="strTrans(this)"><br>
大写金额: <input type="text" name="daxieTxt" size=60 readonly></body>
</html>
```

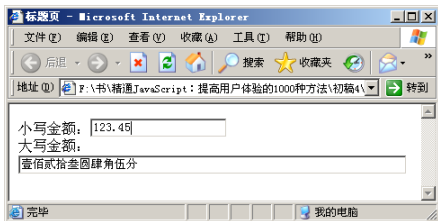


图 14-11 转换为大写金额后的效果

【运行效果】

转换为大写金额后的效果如图 14-11 所示。

【难点剖析】

本例使用“getDaXie”和“millionTrans”方法实现数值型数据的判断，包括如何判断万位数、亿位数等。代码中多次使用正则表达式实现字符的搜索和替换，有关正则表达式的使用，请参考详细资料。

14.21 通过两点坐标计算直线距离

【实例描述】

本例一般用于精确画图，在实际中应用并不多。主要通过给定的两个坐标点计算这两点之间的距离。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function distanceCAL(form)
{
```

```

    var x1 = eval(form.x1.value);           // 获取第一点的 x 坐标
    var y1 = eval(form.y1.value);           // 获取第一点的 y 坐标
    var x2 = eval(form.x2.value);           // 获取第二点的 x 坐标
    var y2 = eval(form.y2.value);           // 获取第二点的 y 坐标
    var calX = x2 - x1;
    var calY = y2 - y1;
    form.result.value = Math.pow((calX * calX + calY * calY), 0.5);
}
</script>
</head>
<body >
<form>
<table border=3 cellspacing=2 cellpadding=5>
<tr>
    <td colspan=4 align=center height="18">
        <strong><span style="font-size: 14pt">两点的直线距离</span></strong></td>
</tr>
<tr>
    <td colspan=2 align=center>
        <span style="color: #666666">第一点的坐标</span></td>
    <td colspan=2 align=center>
        <span style="color: #999966">第二点的坐标</span></td>
</tr>
<tr>
    <td align=center>X 轴</td>
    <td align=center>Y 轴</td>
    <td align=center>X 轴</td>
    <td align=center>Y 轴</td>
</tr>
<tr>
    <td align=center><input type=text name=x1 size=5></td>
    <td align=center><input type=text name=y1 size=5></td>
    <td align=center><input type=text name=x2 size=5></td>
    <td align=center><input type=text name=y2 size=5></td>
</tr>
<tr>
    <td colspan=4 align=center>
        <input type=button value="计算距离" onClick="distanceCAL(this.form)">
        <input type=text name=result size=20></td>
</tr>
</table>
</form>
</body>
</html>

```

【运行效果】

本例的运行效果如图 14-12 所示。

【难点剖析】

本例的重点是如何计算两个指定坐标点之间的距离。获取坐标的值非常容易，使用

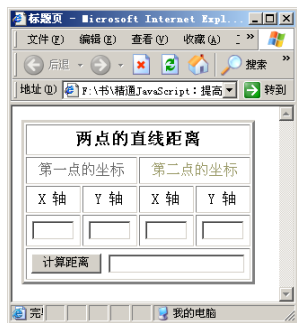


图 14-12 本例的运行效果

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
var oldNum=[]; //彩票中所有的号码
var newNum=[]; //被选中的号码
var num;
for(var i=1;i<30;i++)
    oldNum.push(i<10 ? "0"+i : i)//不足 10 位时，补"0"
for(var i=0;i<7;i++)
{
    num = oldNum.splice(Math.floor(Math.random() * oldNum.length),1);
    newNum.push(num); //选中随机数
                    //添加新元素到数组中
}
alert("抽出号码为： : "+newNum.sort());
</script>
</head>
<body>
</body>
</html>
```

【运行效果】

随机抽取的结果如图 14-13 所示。

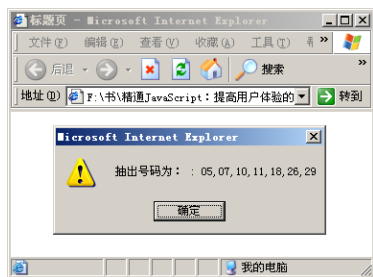


图 14-13 随机抽取的结果

“form.x1.value”即可。计算两个点之间的距离就得使用“Math.pow”方法实现，其使用语法是“Math.pow(x,n)”，返回结果是“x”的“n”次方值。

14.22 随机抽取彩票

【实例描述】

彩票是国家推行的一种福利机制，而彩票机可以提供自动选择号码的功能。本例通过一个简单的方法实现随机号码的选取。

【难点剖析】

本例的重点是随机数的抽取。代码中创建了两个数组，一个用来存放所有号码，一个用来存放抽取出来的号码。抽取号码使用的语句是“Math.random() * oldNum.length”，其中“Math.random”方法用来获取“0~1”之间的任意随机小数。代码中的“splice”方法用来将选中的随机数从当前数组中移除，主要是为了不产生重复的号码。“push”将随机选择号添加到数组中。

14.23 实时计算折扣

【实例描述】

随着网络商店的大量开通，越来越多的人体验了网络购物的方便。在购物时将物品添加到购物车后，系统会根据商品的折扣、价格和数量，自动计算最终付款金额。这就是本例要实现的实时计算功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script type="text/javascript">
var arr = new Array("","","");           //创建数组
function price(myPrice){                  //用数组保存价格信息
    arr[0] = myPrice;
    count();                               //计算结果
}
function Tcount(myCount){

    arr[1] = myCount;                      //用数组保存数量信息
    count();                               //计算结果
}
function off(myOff){
    arr[2] = myOff;                        //保存折扣信息
    count();                               //计算结果
}
function count(){

    myResult.value =arr[0]*arr[1]*arr[2];   //返回计算结果
}
</script>
</head>
<body>
单价: <input type="text" name="PriceObj" onkeyup="price(this.value)"><br><br>
数量: <input type="text" name="CountObj" onkeyup="Tcount(this.value)"><br><br>
折扣: <input type="text" name="OffObj" onkeyup="off(this.value)"><br><br><br>
计算公式: 单价*数量*折扣<br><br>
金额: <input type="text" name="myResult"><br><br>
</body>
</html>
```

【运行效果】

折扣计算效果如图 14-14 所示。

【难点剖析】

本例的重点是实时计算。当用户输入三个条件中的任意一个时，通过“onkeyup”事件获取

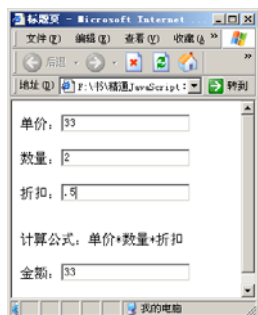


图 14-14 折扣计算效果

用户的输入值，然后在事件中调用“count”方法，根据指定公式立刻计算出结果。

14.24 实用计算器

【实例描述】

JavaScript 虽然是脚本语言，但也可以设计一些常用的工具，如日历、计算器等。本例学习使用 JavaScript 制作计算器。

【实现代码】

```
<script language="JavaScript">
var results='' ; //显示结果
var previouskey='' ; //代表上一个字符
var re=/(\|\/|\*|\+|-)/ //用来判断+-*/的正则
var re2=/(\|\/|\*|\+|-){2}$/ ; //用来判断出现两次+-*/的正则
var re3=/\.\+((\|\/|\*|\+|-)\.+)/ ; //用来判断小数点+-*/的正则
var re4=/\d|\. / ; //用来判断(小数点或数值)的正则
var re5=/^[^\|\/|\*|\+|-]\d$/ ; //用来判断(以+-*/开始)的正则
var re6=/\./ ; //用来判断小数点的正则
//计算结果的方法
function calculate()
{
//判断用户如何输入了一个值，然后单击了"="
if (event.srcElement.tagName=="TD"){
if (event.srcElement.innerText.match(re4)&&previouskey=="")
results='';
if (result.innerText.match(re3)&&event.srcElement.innerText.match(re)){
if (!results.match(re5)){
result.innerText="输入错误!";
return;
}
//以下是要求长度只能保持在 12 位以内(有小数点的情况下)
results=eval(results); //转换为数值
if (results.toString().length>12&&results.toString().match(re6))
results=results.toString().substring(0,12);
result.innerText=results;
}
//出现两次计算符号的情形
results+=event.srcElement.innerText;
if (results.match(re2))
results=results.substring(0,results.length-2)+results.charAt(results.length-1);
result.innerText=results;
}}
function calculateresult()
{
```



```

//当首字符输入错误时
if (!results.match(re5)){
result.innerText="输入错误!";
return;}
results=eval(results);
//转换结果为数值型
if (results.toString().length>=12&&results.toString().match(re6))
results=results.toString().substring(0,12);
result.innerText=results;
}
function pn()
{
//首字符为负数的计算
if (result.innerText.charAt(0)!='-')
result.innerText=results='-'+result.innerText
else if (result.innerText.charAt(0)=='-')
result.innerText=results=result.innerText*(-1)
}
</script>

```

需要在 body 中添加一个 Table，用来呈现计算器的外观，代码如下所示：

```

<body>
<table border="0" cellspacing="0" cellpadding="0" width="400">
<tr>
<td width="100%" valign="top"><table border="2" width="200" cellspacing="0"
cellpadding="0" bgcolor="#000000" style="border-color:black"
onClick="previouskey=event.srcElement.innerText">
<tr>
<td width="100%" bgcolor="#FFFFFF" id="result"
style="font:bold 20px Verdana;color:black;text-align='right'">0</td>
</tr>
<tr>
<td width="100%" valign="middle" align="center"><table border="0" width="100%"
cellspacing="0" cellpadding="0" style="font:bold 20px Verdana;color:white">
<tr><td width="80%" align="center"><table border="1" width="100%" cellspacing="0"
cellpadding="0" style="cursor:hand;font:bold 20px Verdana;color:white"
onmouseover="if (event.srcElement.tagName=='TD')event.srcElement.style. color=
'yellow'"
onmouseout="event.srcElement.style.color='white'" onselectStart="return false"
onClick="calculate()" height="82">
<tr><td width="25%" align="center" height="17">7</td>
<td width="25%" align="center" height="17">8</td>
<td width="25%" align="center" height="17">9</td>
<td width="25%" align="center" height="17">*</td>
</tr><tr><td width="25%" align="center" height="19">4</td>
<td width="25%" align="center" height="19">5</td>
<td width="25%" align="center" height="19">6</td>
<td width="25%" align="center" height="19">*</td>

```



```

</tr><tr><td width="25%" align="center" height="19">1</td>
<td width="25%" align="center" height="19">2</td>
<td width="25%" align="center" height="19">3</td>
<td width="25%" align="center" height="19">-</td>
</tr><tr><td width="25%" align="center" height="19">0</td>
<td width="25%" align="center" height="19"
onClick="pn();previouskey=1;event.cancelBubble=true">+/-</td>
<td width="25%" align="center" height="19">.</td>
<td width="25%" align="center" height="19">+</td>
</tr></table></td><td width="20%"><div align="left"><table border="1" width="100%"
cellspacing="0"
cellpadding="0"><tr><td width="100%" style="cursor:hand;font:bold 20px Verdana;
color:white;text-align:center"
onClick="result.innerText=0;results=''>C</td>
</tr></table></div><div align="left"><table border="1" width="100%" cellspacing="0"
cellpadding="0"
height="81">
<tr><td width="100%" style="cursor:hand;font:bold 32px Verdana;color:white;
text-align:center"
onmouseover="event.srcElement.style.color='yellow'"
onmouseout="event.srcElement.style.color='white'" onClick="calclateresult()">=
</td>
</tr>
</table>
</div>
</div>
</body>

```



图 14-15 计算器的运行界面

【运行效果】

计算器的运行界面如图 14-15 所示。

【难点剖析】

本例的重点在于对用户输入字符的判断。这里要提到的就是正则表达式，其在 JavaScript 中的应用非常灵活，可用来判断用户输入的字符是否合法，也可用来截取页面的内容。本例对使用的正则表达式进行了详细的注释，要了解正则表达式的详细应用语法，可参考相关资料。

14.25 前面补 0 的方法

【实例描述】

在很多数值型和字符型数据中，如果位数不够，通常需要在数据前面补 0 以实现整体效果。本例学习如何实现补 0 操作。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>

```

```

<body >
<div align="center">
<script language="javascript">
function addZero(a,b,c)
{
    while(a<b)                //判断范围
    {
        t=a+"";                //t 为字符型
        while(t.length<c)t="0"+t;    //通过一个循环判断字符的宽度，不够宽度则补 0
        a++;
        document.write(t+"<br>");    //输出补 0 后的数据
    }
}
addZero(1,8,5);
</script>
</div>
</body>
</html>

```

【运行效果】

补 0 效果如图 14-16 所示。

【难点剖析】

本例的重点是字符类型的转换和字符宽度的设置。“t=a+”这句代码是将变量“a”转换为字符型。“length”属性用来判断当前字符串的长度。

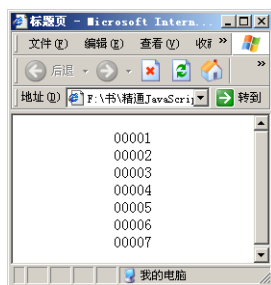


图 14-16 补 0 效果

第 15 章 图片的特效

本章导读

图片是网页中不可缺少的元素，包括网站的 LOGO 图片、新闻图片等。本章不仅包含一些简单的图片特效，如图片翻转、滚动等，还提供一些网站必须的图片操作技巧，如禁止图片复制、图片拖放、图片的等比例缩略图等。

15.1 图片变形效果

【实例描述】

在网页中可以通过 CSS 滤镜改变图片的显示方式。本例学习如何实现图片的变形效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
</head>
<body>
  <p><font face="Arial" color="#008080">
    <div><p align="center">
      
    </p>  </div>    </font>
    <p align="center">这是图片的变形效果</p></font>
  </body>
</html>
```

【运行效果】

变形效果如图 15-1 所示。

【难点剖析】

本例的重点是 CSS 中“filter”滤镜的使用。“wave”属性表示以波浪的形式显示图片，其使用语法如下所示：

```
filter: Wave(Add=add, Freq=freq, LightStrength=strength, Phase=phase, Strength=strength)
```

参数的说明如下：

- Add：一般为 1 或 0；
- Freq：变形值；
- LightStrength：变形百分比；
- Phase：角度变形百分比；
- Strength：变形强度。



图 15-1 变形效果

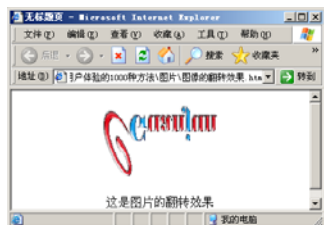
15.2 图片的翻转效果

【实例描述】

图片既可以通过滤镜也可以通过 Photoshop 进行特殊处理。本例介绍如何利用 CSS 的 filter 实现图片翻转。

【实现代码】

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
</head>
<body>
  <p><font face="Arial" color="#008080">
    <div><p align="center">
       </p>
    </div>
  </font>
  <p align="center">这是图片的翻转效果</p></font>
</body>
</html>
```



【运行效果】

图片的翻转效果如图 15-2 所示。

【难点剖析】

本例的重点是 CSS 滤镜“filter:flipV”。“flipV”表示垂直翻转图片，如果修改为“flipHs”则表示水平翻转图片。

图 15-2 图片的翻转效果

15.3 图片的模糊效果

【实例描述】

很多网站中图片的模糊效果使用 Photoshop 完成，其实 CSS 同样提供了实现模糊效果的滤镜。本例学习如何实现图片的模糊效果。

【实现代码】

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
</head>
<body>
  <p><font face="Arial" color="#008080">
    <div><p align="center">
       </p>
    </div>
  </font>
</body>
```

```
<p align="center">这是图片的模糊效果</p></font>
</body>
</html>
```

【运行效果】

图片的模糊效果如图 15-3 所示。

【难点剖析】

本例的重点是 CSS 的滤镜“filter:blur”，其中“blur”的语法如下所示：

```
STYLE="filter:Blur(Add = add, Direction = direction, Strength = strength)"
```

各参数的说明如下：

- Add：一般为 1 或 0；
- Direction：角度，0~315 度，步长为 45 度；
- Strength：效果增长的数值，一般取 5 即可。

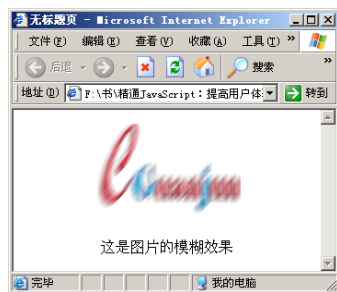


图 15-3 图片的模糊效果

15.4 图片的水印效果

【实例描述】

图片的水印效果通过设置样式的透明度实现。本例以具体实例介绍如何将图片设置为水印样式。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>无标题页</title>
</head>
<body>
  <p><font face="Arial" color="#008080">
    <div><p align="center">
       </p>
    </div>
  </font>
  <p align="center">这是图片的水印效果</p></font>
</body>
</html>
```

【运行效果】

图片的水印效果如图 15-4 所示。

【难点剖析】

本例的重点是 CSS 的“filter”滤镜效果。“alpha”用来设置透明层次，其使用语法如下所示：

```
STYLE="filter:Alpha(Opacity=opacity, FinishOpacity=finishopacity, Style=style, StartX=
startX, StartY=startY, FinishX=finishX, FinishY=finishY)"
```



图 15-4 图片的水印效果

各参数的说明如下：

- Opacity：起始值，取值为 0 ~ 100，0 为透明，100 为原图；
- FinishOpacity：目标值；
- Style：1、2 或 3；
- StartX：任意值；
- StartY：任意值。

15.5 图片淡出淡隐

【实例描述】

图片的透明度可以让图片具有特殊的效果。本例学习如何利用图片的透明特性实现淡出淡隐效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body onLoad="fade()">

<script language="JavaScript">
var bOpa = 1;
var c = true;
function fade()
{
    if(document.all);
    if(c == true) {
        bOpa++;
    }
    if(bOpa==100) {                                     //透明度到 100 时，开始减少
        bOpa--;
        c = false
    }
    if(bOpa==10) {                                     //透明度到 10 时，控制透明度增减的参数变为 true
        bOpa++;
        c = true;
    }
    if(c == false) { //控制透明度是加还是减
        bOpa--;
```



```

    }
    img.filters.alpha.opacity=0 + bOpa;           //改变图片的透明度
    setTimeout("fade()",50);                       //定时执行 fade 方法
}
</script></body>
</html>

```

【运行效果】

图片淡出淡隐的效果如图 15-5 所示。

【难点剖析】

本例的重点是图片的透明度属性“opacity”。此属性在本例中通过 CSS 的滤镜完成，“filters.alpha.opacity”用来设置图片的样式，通过不断增加或减少图片的透明度实现淡出淡隐效果。



图 15-5 图片淡出淡隐的效果

15.6 图片的渐隐播放效果

【实例描述】

单独实现一个图片的显示和隐藏比较容易,本例将介绍如何实现多个图片连续播放的效果,并在图片切换的时候使用逐渐消隐的特效。

【实现代码】

```

<script language=JavaScript>
var strngth=1
var index_image=0
var imageSrc = new Array()
imageSrc[0] = "LOGO1.gif"
imageSrc[1] = "LOGO2.gif"
imageSrc[2] = "LOGO1.gif"
function showimage()                                //显示图片
{
    if(document.all) {
        //创建一个带滤镜样式的 img 图片——注意显示的图片并不固定
        if (strngth <=110) {
            imagediv.innerHTML="<img style='filter:alpha(opacity="+strngth+"' src=
"+imageSrc [index_image]+" border=0>";
            strngth=strngth+10
            var timer=setTimeout("showimage()",100)           //每隔 100 毫秒显示图片
        }
        else {
            clearTimeout(timer)
            var timer=setTimeout("hideimage()",1000)          //每隔 1000 毫秒隐藏图片
        }
    }
    //是 Netscape 浏览器时的特效实现方法
}

```



```
if(document.layers) {
clearTimeout(timer)
document.imagediv.document.write("")
document.close()
index_image++
if (index_image >= imageSrc.length) {index_image=0}
var timer=setTimeout("showimage()",2000)
}
}
function hideimage() //隐藏图片
{
if (strngth >=-10) {
//设置图片逐渐消隐的滤镜效果——注意图片并不固定
imagediv.innerHTML="<img style='filter:alpha(opacity="+strngth+")' src= "+imageSrc
[index_image]+" border=0>";
strngth=strngth-10
var timer=setTimeout("hideimage()",100) //每隔 100 毫秒就隐藏图片
}
else {
clearTimeout(timer)
index_image++
if (index_image >= imageSrc.length) {index_image=0}
strngth=1
var timer=setTimeout("showimage()",500) //每隔 500 毫秒就显示图片
}
}
}</script>
```



图 15-6 图片的渐隐播放效果

【运行效果】

图片的渐隐播放效果如图 15-6 所示。

【难点剖析】

本例的重点是滤镜结合定时器的使用。用滤镜“filter:alpha”实现图片的渐隐渐现，然后使用定时器“setTimeout”实现图片数组的调用。

15.7 文字环绕图片

【实例描述】

在 Word 文档的制作过程中经常会遇到图片和文字都存在的情况，为了版面更合理通常需要特殊的效果，如文字环绕图片。本例就学习如何使文字环绕图片。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
```

```
<title>无标题页</title>
</head>
<body>
<div style="background:#EEEEEE;padding:10px;width:240px;">

<div id="mydiv" style="clear:none;">国家环保总局宣布今起松花江进入为期十年的“休养期”，并提出
70 条具体的治理措施，其中包括停止审批所有从松花江流域向外流域的调水工程，逐步淘汰松花江流域钢铁、电力、焦炭、
造纸等行业的落后生产能力。</div>
</body>
</html>
```

【运行效果】

文字环绕图片的效果如图 15-7 所示。

【难点剖析】

本例的重点是样式表中的“clear”。在一个大容器内如果定义了 float 元素（本例中 img 图片定义了 float 样式），那么后面的元素会与其一起分享大容器剩下的宽度，如果既不想和 float 一起同行显示，又不想在浏览器中错位，那就得用到“clear:both”取消所有浮动。

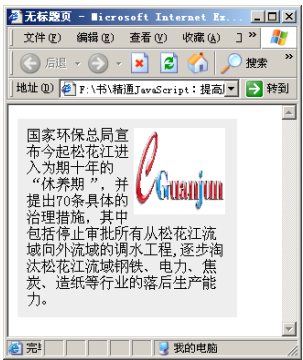


图 15-7 文字环绕图片的效果

15.8 图片切换的特殊效果

【实例描述】

为了增加页面的美观性，在进行图片切换时通常使用类似幻灯片的效果。本例介绍如何实现这种图片切换的特殊效果。

【实现代码】

```
<script language="javascript">
var imgUrl=new Array();
var imgLink=new Array();
var adNum=0;
var jumpUrl="http://pic.music.tom.com/view/35611/tid-35611-pid-1.html";
imgLink[1]="http://pic.music.tom.com/view/35611";
imgLink[2]="http://pic.music.tom.com";
imgUrl[1]="http://cimg.163.com/tech/2006/1/18/2006011810122068706.jpg";
imgUrl[2]="http://cimg.163.com/tech/2006/1/17/200601171002503f251.jpg";
var imgPre=new Array();
var j=0;
for (i=1;i<=imgUrl.length-1;i++)
{
if(imgLink[i]!="") {j++;}
else {break;}
}
function playTran()
```



```
{
    if (document.all)
        imgInit.filters.revealTrans.play();
}
var key=0;
//循环显示图片
function nextAd()
{
    if(adNum<j)adNum++ ;
    else adNum=1;
    if( key==0 ){key=1;}
    else if (document.all){
        imgInit.filters.revealTrans.Transition=6;
        imgInit.filters.revealTrans.apply();
        playTran();
    }
    document.images.imgInit.src=imgUrl[adNum];
    jumpUrl=imgLink[adNum];
    theTimer=setTimeout("nextAd()", 7000);
}
//单击图片时实现导航
function goUrl()
{
    if (jumpUrl != ''){
        if (jumpTarget != '') window.open(jumpUrl,'_blank');
        else location.href=jumpUrl;
    }
}
}</script>
```

需要在 body 中添加一个链接，代码如下所示：

```
<a href="javascript:goUrl()"></a>
```



图 15-8 实例运行界面

【运行效果】

实例运行界面如图 15-8 所示。

【难点剖析】

本例的重点是调用了 CSS 中的滤镜“revealTrans”，其使用格式如下所示：

Filter:revealtrans(duration=转换的秒数, transition=转换的类型);

“transition”参数表示图片切换的样式，如可以是卷帘、溶解等效果，其取值局限于 1~23 之间。其中各值所代表的样式如表 15-1 所示。

表 15-1 切换样式表

样 式	值	样 式	值
矩形从大到小	0	随机溶解	12
矩形从小到大	1	垂直向内裂开	13
圆形从大到小	2	垂直向外裂开	14
圆形从小到大	3	水平向内裂开	15
向上推开	4	水平向外裂开	16
向下推开	5	向左下剥开	17
向右推开	6	向左上剥开	18
向左推开	7	向右下剥开	19
垂直型百叶窗	8	向右上剥开	20
水平型百叶窗	9	随机水平细纹	21
水平棋盘	10	随机垂直细纹	22
垂直棋盘	11	随机选取一种特效	23

15.9 晃动的图片

【实例描述】

文字可以实现左右滚动，图片也可以实现左右移动。本例介绍一个图片左右移动的特效。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
step = 0;
obj = new Image();           //创建图片对象
function anim(xp,xk,smer)    //smer 代表晃动方向
{
    obj.style.left = x;
    x += step*smer;
    if (x>=(xk+xp)/2) {
        if (smer == 1) step--;           //往左移动
        else step++;
    }
    else {
        if (smer == 1) step++;           //往右移动
        else step--;
    }

    if (x >= xk) {                   //如果已经到右边界，则反向晃动
```



```

        x = xk;
        smer = -1;
    }

    if (x <= xp) {                                //如果已经到左边界，则反向晃动
        x = xp;
        smer = 1;
    }
    setTimeout('anim('+xp+', '+xk+', '+smer+')', 40); //设置定时器，实现不断晃动效果
}

function moveLR(objID,movingarea_width,c)
{
    if (navigator.appName=="Netscape") window_width = window.innerWidth;
    else window_width = document.body.offsetWidth;    //获取窗体的宽度
    obj = document.images[objID];
    image_width = obj.width;                          //获取图像的宽度

    x1 = obj.style.left;                              //获取图像的 x 坐标
    x = Number(x1.substring(0,x1.length-2));          //去掉后面的像素标记“px”
    if (c == 0) {
        if (movingarea_width == 0) {                  //没有设置移动区域的情况
            right_margin = window_width - image_width;
            anim(x,right_margin,1);                    //开始晃动图片
        }
        else {
            right_margin = x + movingarea_width - image_width;
            if (movingarea_width < x + image_width) window.alert("No space for moving!");
            else anim(x,right_margin,1);
        }
    }
    else {
        if (movingarea_width == 0)                    //没有设置移动区域的情况
            right_margin = window_width - image_width;
        else {
            x = Math.round((window_width-movingarea_width)/2);
            right_margin = Math.round((window_width+movingarea_width)/2)-image_width;
            //获取可以移动的空间
        }
        anim(x,right_margin,1);
    }
}
</script>

<script language="JavaScript">
    setTimeout("moveLR('myImg',400,1)",10);

```

```

</script>
</head>
<body>
</body>
</html>

```

【难点剖析】

本例的重点是定时器和随机数的应用。定时器用来不断移动图片，随机数的获取是使用“Math.random”函数。“Math.round”是四舍五入函数，返回一个伪随机数（0~1 之间的 double 数）。

15.10 定时消失的图片

【实例描述】

为了不用经常更新网站，可以为一些图片和文本设置有效期，有效期过后这些图片或文本自动消失。本例通过 JavaScript 提供的“Date”对象，实现图片的定时消失。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function view(cDays, cDate)
{
    var oldDate = new Date(cDate);           //将指定的日期转换为标准日期型
    var newDate = new Date();                 //获取当天的日期
    var maxtime = cDays*24*60*60*1000;        //将指定日期换算为毫秒
    var psj=0;
    if ((newDate.getTime()-oldDate.getTime()) <= maxtime) //判断日期是否还在指定时间内
    {
        //动态显示指定的图片
        document.write("<img src='http://www.google.cn/intl/zh-CN/images/ logo_cn. gif'
width=400 height=100 ALIGN=MIDDLE Alt='myimg'>");
    }
}
view(7,"05/31/2008")                        //调用图片定时消失的方法
</script>
</head>
<body>
</body>
</html>

```

【难点剖析】

本例的重点是 JavaScript 的 Date 对象。“new Date”用来创建一个新的日期对象，也就是当前日期。而“new Date（指定日期）”则是将指定的日期转换为日期对象。“getTime”方法返回

一个整数值，此值是一个单位是毫秒的时间间隔。这段时间间隔是从 1970 年 1 月 1 日到指定的日期的时间。

15.11 QQ 图片一闪一闪的效果

【实例描述】

QQ 是一个比较流行的聊天软件，当好友说话时其头像会一闪一闪。本例使用一个非常简便的方法实现类似的特效。

【实现代码】

```
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=gb2312">
<TITLE>无标题文档</TITLE>
</HEAD>
<BODY>
<IMG id="imgId" style="visibility:visible " src="your.gif">
<SCRIPT language="javascript">
function showImg()
{
    if(imgId.style.visibility == "visible")           //如果可见，则隐藏
        imgId.style.visibility = "hidden";
    else
        imgId.style.visibility = "visible";           //设置图片可见
    setTimeout('showImg()',300);                       //间隔的毫秒
}
showImg();
</SCRIPT>
</BODY>
</HTML>
```



图 15-9 本例的运行效果

【运行效果】

本例的运行效果如图 15-9 所示。

【难点剖析】

本例通过一个定时器每隔 300 毫秒轮换图片的显示状态，使其一直处于显示和隐藏状态的不断交替中，以此实现一闪一闪的效果。

15.12 设置 textarea 中的图片不处于编辑状态

【实例描述】

有时需要在文本区域内插入图片，类似在线编辑器的效果。当图片显示在文本区域后，如

果用鼠标选中图片则会出现类似于编辑效果的 8 个边角。本例学习如何去掉这些边角，使图片处于不可编辑状态。这在设计在线编辑器时非常有效。

【实现代码】

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page</title>
</head>
<body>
<textarea id=t1 cols=30 rows=3 ></textarea>

<script language="javascript">
t1.appendChild(img1)                //动态添加图片到文本区域
img1.UNSELECTABLE=true;             //取消图片的选中操作
</script>
</body>
</html>
```

【运行效果】

本例的运行效果如图 15-10 所示。

【难点剖析】

本例的难点有两个：动态在文本区域中添加图片标签、设置图片的选中状态。默认情况下不允许将 `img` 标签直接写到 `textarea` 标签内，所以本例使用“`appendChild`”方法实现动态添加 `img` 元素。“`UNSELECTABLE`”属性用来指定当前元素不可被选中。

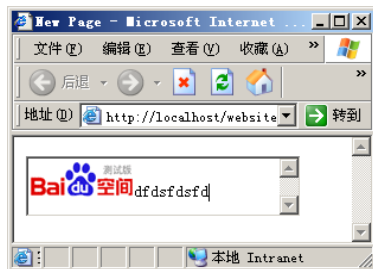


图 15-10 本例的运行效果

15.13 禁止图片的复制

【实例描述】

有时为了保护图片的版权，页面中的图片是不允许复制的。本例介绍使用一个方法屏蔽对网页中所有图片的复制操作。

【实现代码】

```
function noCopy(control)
{
alert("版权所有，禁止复制！");
return false;
}
function check()
{
if(document.images) //遍历页面中的图片
```

```
for(i=0;i<document.images.length;i++)  
    document.images[i].onmousedown = noCopy;  
}  
</script>
```

需要在 body 中添加事件 onload，代码如下所示。

```
<body onload="check()">
```

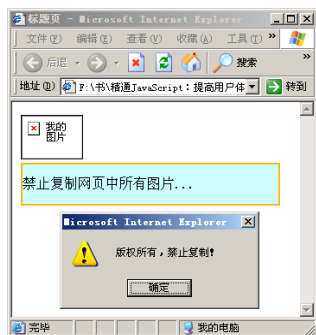


图 15-11 禁止复制图片的提示

【运行效果】

禁止复制图片的提示如图 15-11 所示。

【难点剖析】

本例不是通过屏蔽鼠标右键来完成禁止功能，而是通过遍历页面中所有的图片，并为每个图片添加“onmousedown”事件，此事件包含一个提示功能，并返回“false”值以达到禁止复制的效果。

15.14 LOGO 像雪花一样落下

【实例描述】

在页面中可以经常看到一些像雪花一样落下的文本、广告、图片等。本例将以具体实例演示如何设计像雪花一样落下的 LOGO。

【实现代码】

```
<SCRIPT language=JavaScript>  
//判断浏览器  
var ns4up = (document.layers) ? 1 : 0;  
var ie4up = (document.all) ? 1 : 0;  
var no = 3; //显示的图片数量  
var speed = 20; //图片移动的速度  
var snowflake = "LOGO1.gif"; //图片地址  
var filen = "http://www.baidu.com" //单击图片时的链接  
var dx, xp, yp;  
var am, stx, sty;  
var i, doc_width = 400, doc_height = 600;  
if (ns4up) {  
    doc_width = self.innerWidth;  
    doc_height = self.innerHeight;  
} else if (ie4up) {  
    doc_width = document.body.clientWidth;  
    doc_height = document.body.clientHeight;  
}  
dx = new Array();  
xp = new Array();  
yp = new Array();  
am = new Array();
```

```

stx = new Array();
sty = new Array();
//以图片的数量为循环，一次设置每个图片的位置和速度等。
for (i = 0; i < no; ++ i)
{
dx[i] = 0;
xp[i] = Math.random()*(doc_width-50);
yp[i] = Math.random()*doc_height;
am[i] = Math.random()*20;
stx[i] = 0.02 + Math.random()/10;
sty[i] = 0.7 + Math.random();
if (ns4up) {
if (i == 0) {
document.write("<html><title></title><body>");
document.write("<layer name=\"dot"+ i +"\" left=\"15\\\" ");
document.write("top=\"15\\\" visibility=\"show\\\"><A href=\""+ filen + "\" target=
\"_blank\\\"><img src=\"");
document.write(snowflake + "\" border=\"0\\\"></layer>");
document.write("</body></html>");
} else {
document.write("<html><title></title><body>");
document.write("<layer name=\"dot"+ i +"\" left=\"15\\\" ");
document.write("top=\"15\\\" visibility=\"show\\\"><A href=\""+ filen + "\" target=
\"_blank\\\"><img src=\"");
document.write(snowflake + "\" border=\"0\\\"></layer>");
document.write("</body></html>");
}
} else if (ie4up) {
if (i == 0) {
document.write("<html><title></title><body>");
document.write("<div id=\"dot"+ i +"\" style=\"POSITION: ";
document.write("absolute; Z-INDEX: "+ i +"; VISIBILITY: ");
document.write("visible; TOP: 15px; LEFT: 15px;\\\"><a href=\""+filen+"\" target=
\"_blank\\\"><img src=\"");
document.write(snowflake + "\" border=\"0\\\"></div>");
document.write("</body></html>");
} else {
document.write("<html><title></title><body>");
document.write("<div id=\"dot"+ i +"\" style=\"POSITION: ");
document.write("absolute; Z-INDEX: "+ i +"; VISIBILITY: ");
document.write("visible; TOP: 15px; LEFT: 15px;\\\"><a href=\""+filen+"\" target=
\"_blank\\\"><img src=\"");
document.write(snowflake + "\" border=\"0\\\"></div>");
document.write("</body></html>");
}
}
}
}
}

```

//实现 Netscape 浏览器中动画效果的方法



```

function snowNS() {
    for (i = 0; i < no; ++ i)
    {
        yp[i] += sty[i];
        if (yp[i] > doc_height-50) {
            xp[i] = Math.random()*(doc_width-am[i]-30);
            yp[i] = 0;
            stx[i] = 0.02 + Math.random()/10;
            sty[i] = 0.7 + Math.random();
            doc_width = self.innerWidth;
            doc_height = self.innerHeight;
        }
        dx[i] += stx[i];
        //指定图片的显示位置: X 和 Y 坐标
        document.layers["dot"+i].top = yp[i];
        document.layers["dot"+i].left = xp[i] + am[i]*Math.sin(dx[i]);
    }
    setTimeout("snowNS()", speed);
}
//实现 IE 浏览器中动画效果的方法
function snowIE()
{
    //判断图片的数量

    for (i = 0; i < no; ++ i) {
        yp[i] += sty[i];
        if (yp[i] > doc_height-50) {
            xp[i] = Math.random()*(doc_width-am[i]-30);
            yp[i] = 0;
            stx[i] = 0.02 + Math.random()/10;
            sty[i] = 0.7 + Math.random();
            //重新获取文档的高度和宽度
            doc_width = document.body.clientWidth;
            doc_height = document.body.clientHeight;
        }
        dx[i] += stx[i];
        //指定图片的显示位置: X 和 Y 坐标
        document.all["dot"+i].style.pixelTop = yp[i];
        document.all["dot"+i].style.pixelLeft = xp[i] + am[i]*Math.sin(dx[i]);
    }
    setTimeout("snowIE()", speed); //根据速度循环执行动画方法
}
//判断浏览器类型, 并调用不同的动画方法
if (ns4up) {
    snowNS();
} else if (ie4up) {
    snowIE();
}
</SCRIPT>

```

【运行效果】

像雪花一样落下的效果如图 15-12 所示。

【难点剖析】

本例的重点是如何动态添加 div 并在 div 中加载 LOGO 图片,同时还要设置这些 LOGO 图片的位置、显示速度等。设置落下的 LOGO 数量可修改“no”变量。

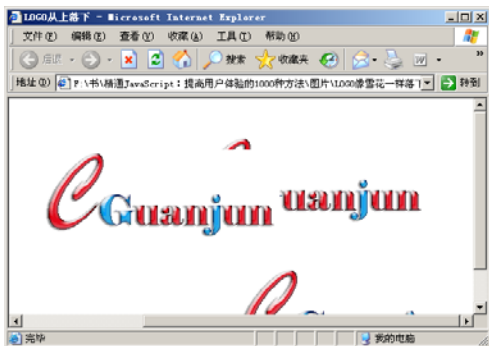


图 15-12 像雪花一样落下的效果

15.15 多幅图片分页滚动显示

【实例描述】

本例可用于图片新闻的展示,通过两个 div 控件循环显示所有的图片。可修改本例中的代码调整图片滚动的速度。

【实现代码】

```
<SCRIPT language="JavaScript">
var scrollerwidth=90                                //指定 div 的高度
var scrollerheight=32                               //指定 div 的宽度
var scrollerbgcolor='white'
var pausebetweenimages=3000                        //图片的间隔时间,默认为三秒
//带链接的图片,存放在数组中
var slideimages=new Array()
slideimages[0]='<A href="http://www.baidu.com" target=_blank><IMG alt=百度搜索 border=0
height=40 src=http://www.baidu.com/img/logo.gif width=100></A>'
slideimages[1]='<A href=http://www.google.cn target=_blank><IMG alt=google 搜索 border=0
height=40 src=http://www.google.cn/intl/zh-CN/images/logo_cn.gif width=100></A>'
slideimages[2]='<A href=# target=_blank><IMG alt=广告位置为你准备 border=0 height=40 src=""
width=100></A>'
slideimages[3]='<A href=http://www.google.cn target=_blank><IMG alt=google 搜索 border=0
height=40 src=http://www.google.cn/intl/zh-CN/images/logo_cn.gif width=100></A>'
if (slideimages.length>1)
i=2                                                    //初始化一个变量,用来做图片数组的索引
else
i=0
function move1(mydiv)
{
    tdiv=eval(mydiv)                                //获取 div 对象
    if (tdiv.style.pixelTop>0&&tdiv.style.pixelTop<=4){ //判断 div 的 y 坐标
        tdiv.style.pixelTop=0                        //指定 div 的 y 坐标
        setTimeout("move1(tdiv)",pausebetweenimages) //设置转换的时间间隔
        setTimeout("move2(secondDiv)",pausebetweenimages)
        return
    }
}
if (tdiv.style.pixelTop>=tdiv.offsetHeight*-1){
```



```

        tdiv.style.pixelTop-=5 //div 开始往上滚动
        setTimeout("move1(tdiv)",100)
    }
    else{
        tdiv.style.pixelTop=scrollerheight //指定 div 的高度
        tdiv.innerHTML=slideimages[i] //将图片显示在 div 中
        if (i==slideimages.length-1 ) //如果已经循环到底，再从第一张开始循环
            i=0
        else
            i++
    }
}
function move2(mydiv)
{
    tdiv2=eval(mydiv) //获取第二个 div
    if (tdiv2.style.pixelTop>0&&tdiv2.style.pixelTop<=4){ //判断 div 的 Y 坐标
        tdiv2.style.pixelTop=0 //指定 div 的 Y 坐标
        setTimeout("move2(tdiv2)",pausebetweenimages) //设置转换的时间间隔
        setTimeout("move1(firstDiv)",pausebetweenimages)
        return
    }
    if (tdiv2.style.pixelTop>=tdiv2.offsetHeight*-1){
        tdiv2.style.pixelTop-=5 //第二个 div 开始向上滚动
        setTimeout("move2(secondDiv)",100)
    }
    else{
        tdiv2.style.pixelTop=scrollerheight //指定第二个 div 的高度
        tdiv2.innerHTML=slideimages[i] //将图片显示在 div 中
        if (i==slideimages.length -1) //如果已经循环到底，再从第一张开始循环
            i=0
        else
            i++
    }
}
function startscroll() //调用实现 div 层移动的方法
{
    if (document.all){
        move1(firstDiv)
        secondDiv.style.top=scrollerheight //设置第二张图片的位置
    }
}

window.onload=startscroll //窗体一加载便开始显示图片
</SCRIPT>

```

【难点剖析】

本例重点是使用两个 div 循环显示所有的图片。为了屏蔽 div 的图层，需要将窗体的背景

色设置为“white”（白色）。然后将第一个 div 的 Y 坐标设置为“1”，指定时间过后再让此 div 的 y 坐标自动减小，实现图层上移的效果。然后设置第二个 div 的 y 坐标，实现第二张图片的显示，依次循环显示所有的图片。

15.16 循环滚动显示图片

【实例描述】

随着 Blog 的日渐鼎盛，个人信息的网络存储越来越受人们青睐。目前最流行的个人相册，是个人风采的又一展示方式。在相册中通常可以通过滚动的形式浏览照片。本例介绍如何在网站中实现图片的滚动。

【实现代码】

```
<script language="javascript">
//定义要显示的图片数组
imgArr=new Array()
imgArr[0]="<a href=http://www.google.com/ onmouseMove='javascript:outHover= true'
onMouseover='javascript:outHover=true' onMouseout='javascript:outHover= false;mvStart()''> <img
src=LOGO1.gif border=0></a>"
imgArr[1]="<a href=http://www.baidu.com/ onmouseMove='javascript:outHover= true'
onMouseover='javascript:outHover=true' onMouseout='javascript:outHover= false;mvStart()''> <img
src=LOGO2.gif border=0></a>"
imgArr[2]="<a href=http://www.google.com/ onmouseMove='javascript:outHover= true'
onMouseover='javascript:outHover=true' onMouseout='javascript:outHover= false;mvStart()''> <img
src=LOGO1.gif border=0></a>"
imgArr[3]="<a href=http://www.baidu.com/ onmouseMove='javascript:outHover= true'
onMouseover='javascript:outHover=true' onMouseout='javascript:outHover= false;mvStart()''> <img
src=LOGO2.gif border=0></a>"
imgArr[4]="<a href=http://www.google.com/ onmouseMove='javascript:outHover= true'
onMouseover='javascript:outHover=true' onMouseout='javascript:outHover= false;mvStart()''> <img
src=LOGO1.gif border=0></a>"
//内部变量
var moveStep=4; //步长, 单位: pixel
var moveRelax=100; //移动时间间隔, 单位: ms
ns4=(document.layers)?true:false;
var displayImgAmount=4 ; //视区窗口可显示个数
var divWidth=220; //每块图片占位宽
var divHeight=145; //每块图片占位高
var startDnum=0;
var nextDnum=startDnum+displayImgAmount;
var timeID;
var outHover=false;
var startDivClipLeft;
var nextDivClipRight;
//初始化层
function initDivPlace()
```



```

{
    if (ns4)
    {
        for (i=0;i<displayImgAmount;i++){
            eval("document.divOuter.document.divAds"+i+".left="+divWidth*i)
        }
        for (i=displayImgAmount;i<imgArr.length;i++){
            eval("document.divOuter.document.divAds"+i+".left="+divWidth* display
ImgAmount)
        }
    }else{
        for (i=0;i<displayImgAmount;i++){
            eval("document.all.divAds"+i+".style.left="+divWidth*i)
        }
        for (i=displayImgAmount;i<imgArr.length;i++){
            eval("document.all.divAds"+i+".style.left="+divWidth*displayImgAmount)
        }
    }
}
//设置定时器移动图片
function mvStart(){
    timeID=setTimeout(moveLeftDiv,moveRelax)
}
//清除定时器，停止移动
function mvStop(){
    clearTimeout(timeID)
}

function moveLeftDiv(){
    if (ns4){
        for (i=0;i<=displayImgAmount;i++){
            eval("document.divOuter.document.divAds"+parseInt((startDnum+i)
%imgArr.length)+" .left=document.divOuter.document.divAds"+parseInt((startDnum+i)%imgArr.l
ength)+" .left-moveStep")
        }

        startDivClipLeft=parseInt(eval("document.divOuter.document.divAds" +startDnum+
".clip.left"))
        nextDivClipRight=parseInt(eval("document.divOuter.document.divAds" +nextDnum+
".clip.right"))

        if (startDivClipLeft+moveStep>divWidth){
            eval("document.divOuter.document.divAds"+nextDnum+".clip.right=" +divWidth)

            eval("document.divOuter.document.divAds"+startDnum+".left=" +divWidth*
displayImgAmount)
            eval("document.divOuter.document.divAds"+parseInt((nextDnum+1) %imgArr.
length)+" .left=document.divOuter.document.divAds"+nextDnum+".left="+divWidth)

```



```

eval("document.divOuter.document.divAds"+parseInt((nextDnum+1) %imgArr.
length)+".clip.left=0")

startDnum=(++startDnum)%imgArr.length
nextDnum=(startDnum+displayImgAmount)%imgArr.length

startDivClipLeft=moveStep-(divWidth-startDivClipLeft)
nextDivClipRight=moveStep-(divWidth-nextDivClipRight)
}else{
eval("document.divOuter.document.divAds"+nextDnum+".clip.left=0")
startDivClipLeft+=moveStep
nextDivClipRight+=moveStep
}
eval("document.divOuter.document.divAds"+startDnum+".clip.left= "+startDivClipLeft)
eval("document.divOuter.document.divAds"+nextDnum+".clip.right= "+nextDivClipRight)
}else{
for (i=0;i<=displayImgAmount;i++){
eval("document.all.divAds"+parseInt((startDnum+i)%imgArr.length)+ ".style.
left=document.all.divAds"+parseInt((startDnum+i)%imgArr.length)+".style.pixelLeft-moveStep")
}

startDivClipLeft=parseInt(eval("document.all.divAds"+startDnum+ ".currentStyle.
clipLeft"))
nextDivClipRight=parseInt(eval("document.all.divAds"+nextDnum+ ".currentStyle.
clipRight"))

if (startDivClipLeft+moveStep>divWidth){
eval("document.all.divAds"+nextDnum+".style.clip='rect(0,"+divWidth+",
"+divHeight+",0"+"'")'")

eval("document.all.divAds"+startDnum+".style.left="+divWidth*
displayImgAmount)
eval("document.all.divAds"+parseInt((nextDnum+1)%imgArr.length)+ ".style.
left=document.all.divAds"+nextDnum+".style.pixelLeft+"+divWidth)

startDnum=(++startDnum)%imgArr.length
nextDnum=(startDnum+displayImgAmount)%imgArr.length

startDivClipLeft=moveStep-(divWidth-startDivClipLeft)
nextDivClipRight=moveStep-(divWidth-nextDivClipRight)
}else{
startDivClipLeft+=moveStep
nextDivClipRight+=moveStep
}
eval("document.all.divAds"+startDnum+".style.clip='rect(0,"+divWidth+",
"+divHeight+", "+startDivClipLeft+"')'")
eval("document.all.divAds"+nextDnum+".style.clip='rect(0,"+nextDivClipRight+",

```



```

"+divHeight+",0)')")
    }
    if (outHover){
        mvStop()
    }else{
        mvStart()
    }
}
//定义显示图片的层
function writeDivs(){
    if (ns4){
        document.write("<ilayer name=divOuter width=750 height="+divHeight+">")

        for (i=0;i<imgArr.length;i++){
            document.write("<layer name=divAds"+i+">")
            document.write(imgArr[i]+" ")
            document.write("</layer>")
        }
        document.write("</ilayer>")
        document.close()
        for (i=displayImgAmount;i<imgArr.length;i++){
            eval("document.divOuter.document.divAds"+i+".clip.right=0")
        }
    }else{
        document.write("<div id=divOuter style='position:relative' width=750 height="+divHeight+">")

        for (i=0;i<imgArr.length;i++){
            document.write("<div id=divAds"+i+" style='position:absolute; clip:rect(0,\""+divWidth+\",\""+divHeight+\",0)'\>")
            document.write(imgArr[i]+" ")
            document.write("</div>")
        }
        document.write("</div>")
        for (i=displayImgAmount;i<imgArr.length;i++){
            eval("document.all.divAds"+i+".style.clip='rect(0,0,\""+divHeight+\",0)'\")
        }
    }
}
writeDivs();
initDivPlace();
</script>

```

需要在 body 中添加启动事件，代码如下所示：

```
<body onload="mvStart()">
```

【运行效果】

滚动图片的显示效果如图 15-13 所示。

【难点剖析】

本例的重点是 `ilayer` 层和 `Array` 数组。其中数组是 JavaScript 的重要对象，用来存储一系列类型相同的数据。`ilayer` 层具有三维的感觉，使设计者能够对相互重叠的层组成的三维层进行精确地控制，这些相互重叠的层是 Web 页上透明或不透明的内容。

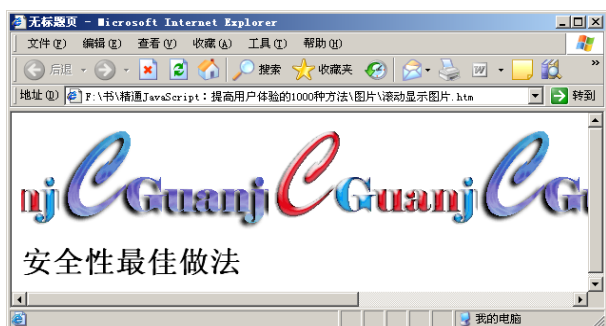


图 15-13 滚动图片的显示效果

15.17 图片的选择展示

【实例描述】

在相册、付款或新闻页面中经常需要用户进行选择，并在选择的同时展示图片界面以确认用户的选择。本例学习使用下拉框实现图片选择展示的效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
</head>
<body>
  <CENTER>
    
  </CENTER>
  <CENTER>
    <SELECT onChange="document.img1.src=options[selectedIndex].value">
      <option value="http://www.google.cn/intl/zh-CN/images/logo_cn.gif">goole 搜索</option>
      <option value="http://www.baidu.com/img/logo.gif">百度搜索</option>
      <option value="http://search.cn.yimg.com/i/070420/lg.gif">雅虎搜索</option>
    </SELECT>
  </CENTER>
</body>
</html>
```

【运行效果】

选择展示后的效果如图 15-14 所示。

【难点剖析】



图 15-14 选择展示后的效果

本例提供了 3 个搜索 LOGO 的选择，重点是 select 控件的“onChange”事件。当用户选择下拉框中的数据时，根据用户选择值的索引“options[selectedIndex]”，用 img 控件显示用户选择的 LOGO。

15.18 图片新闻切换效果

【实例描述】

新闻切换技术想必大家都很熟悉，本例研究其切换效果如何实现。

【实现代码】

```
<script language="javascript">
var nn;
nn=1;
setTimeout('change_img()',6000);
function change_img() //更改图片
{
    if(nn>4) nn=1
    setTimeout('setFocus1('+nn+')',6000);
    nn++;
    tt=setTimeout('change_img()',6000);
}
function setFocus1(i) //选择层
{
    selectLayer1(i);
}
//判断显示的层
function selectLayer1(i)
{
    switch(i)
    {
        case 1:
            document.getElementById("focusPic1").style.display="block";
            document.getElementById("focusPic2").style.display="none";
            document.getElementById("focusPic3").style.display="none";
            document.getElementById("focusPic4").style.display="none";
            document.getElementById("focusPic1nav").style.display="block";
            document.getElementById("focusPic2nav").style.display="none";
            document.getElementById("focusPic3nav").style.display="none";
            document.getElementById("focusPic4nav").style.display="none";
            break;
        case 2:
            document.getElementById("focusPic1").style.display="none";
            document.getElementById("focusPic2").style.display="block";
```

```

document.getElementById("focusPic3").style.display="none";
document.getElementById("focusPic4").style.display="none";
document.getElementById("focusPic1nav").style.display="none";
document.getElementById("focusPic2nav").style.display="block";
document.getElementById("focusPic3nav").style.display="none";
document.getElementById("focusPic4nav").style.display="none";
break;
case 3:
document.getElementById("focusPic1").style.display="none";
document.getElementById("focusPic2").style.display="none";
document.getElementById("focusPic3").style.display="block";
document.getElementById("focusPic4").style.display="none";
document.getElementById("focusPic1nav").style.display="none";
document.getElementById("focusPic2nav").style.display="none";
document.getElementById("focusPic3nav").style.display="block";
document.getElementById("focusPic4nav").style.display="none";
break;
case 4:
document.getElementById("focusPic1").style.display="none";
document.getElementById("focusPic2").style.display="none";
document.getElementById("focusPic3").style.display="none";
document.getElementById("focusPic4").style.display="block";
document.getElementById("focusPic1nav").style.display="none";
document.getElementById("focusPic2nav").style.display="none";
document.getElementById("focusPic3nav").style.display="none";
document.getElementById("focusPic4nav").style.display="block";
break;
}
}
</script>

```

需要在 body 中添加一些新闻，注意这些新闻的布局。由于代码比较多，请参考随书光盘，并且要注意其中使用的样式。

【运行效果】

页面运行效果如图 15-15 所示。

【难点剖析】

本例的重点是页面的布局和控制件的样式。其中要注意获取页面元素使用的是“document.getElementById()”方法，其中的参数就是要获取的元素 id。修改元素的样式使用“style”属性，其中“style.display”用来控制元素的显示和隐藏。



图 15-15 图片新闻切换效果

15.19 判断上传图片的大小

【实例描述】

上传文件或图片的大小都会受到限制。本例学习如何在客户端判断用户上传图片的大小。



【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function Judge()
{
    //判断图片的大小是否大于 50，或者小于 0（没有选择图片）
    if(document.all.myImg.fileSize>1024*50||document.all.myImg.fileSize<=0)
    {
        alert('请选择小于 50K 的图片!');
        return false;                                //不执行任何操作
    }
    else{
        alert('上传没问题');
    }
}
</script>
</head>
<body>
<input type="file" id="jia" onchange="document.all.myImg.src=this.value"/>
<img src="" id="myImg" style="display:none">
<button onclick="Judge()">上传图片</button>
</body>
</html>
```

【难点剖析】

本例的重点是使用 file 控件和 img 控件。file 控件用来上传文件，img 用来显示上传的图片。通过 file 控件的“onchange”事件获取图片。img 控件提供一个“fileSize”属性来获取图片的大小。

15.20 上传图片时预览

【实例描述】

为了让用户确认选择的图片是否正确，网页可以提供客户端的图片预览功能。本例学习如何预览图片。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
</head>
<body>
<form name="form1" id="form1" method="post" action="#">
<input type="file" name="file1" id="file1" onchange="preview()"/>
```

```




</form>
<script type="text/javascript">
function preview()
{
var x = document.getElementById("file1");           //获取上传控件
if(!x || !x.value) return;
var patn = /\.jpg$|\.jpeg$|\.gif$/i;                //使用正则判断用户选择的文件类型
if(patn.test(x.value)){
var y = document.getElementById("img1");           //获取图片控件
if(y){ y.src = 'file://localhost/' + x.value; }      //获取图片源
else{ var img=document.createElement('img');        //创建图片文件，并设置图片的高度、宽度和id
img.setAttribute('src','file://localhost/'+x.value);
img.setAttribute('width','120');
img.setAttribute('height','90');
img.setAttribute('id','img1');
document.getElementById('form1').appendChild(img); } }
else{ alert("您选择的不是图片文件。"); } }
</script>
</body>
</html>

```

【运行效果】

图片预览效果如图 15-16 所示。

【难点剖析】

本例的重点是如何动态创建一个图片，并显示指定的图片。创建使用的是 DOM 对象的“createElement”方法，指定图片使用的是控件的“src”属性。本例还有一个难点是如何判断用户选择的文件类型，这里使用了正则表达式，具体正则表达式的使用方法用户可参考相关资料。

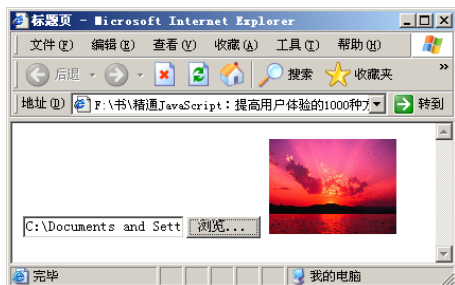


图 15-16 图片预览效果

15.21 对联广告

【实例描述】

大型门户网站为了获取利益都提供了大量的广告。对联广告就是常用的具有中国特色的广告。本例学习制作对联广告。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=gb2312" />
<title></title>
</head>

```



```
<body leftmargin="0" topmargin="0">
<script type="text/javascript">
    var delta=0.115
    var collection;
    function floaters()
    {
        this.items = [];
        //在页面中动态添加div, 参数依次代表: div 的 id, X 坐标,Y 坐标,显示的内容
        this.addItem= function(id,x,y,content)
        {
            document.write('<DIV id='+id+' style="Z-INDEX: 0; POSITION: absolute;
width:80px; height:60px;left:'+(typeof(x)=='string'?eval(x):x)+'; top:'+(typeof(y)=='
'string'?eval(y):y)+'">'+content+'</DIV>');
            var newItem = {};
            newItem.object = document.getElementById(id);
            newItem.x = x;
            newItem.y = y;
            this.items[this.items.length] = newItem;
        }
        this.play= function()
        {
            collection = this.items
            setInterval('play()',10);
        }
    }
    //显示对联, 此方法绑定到定时器
    function play()
    {
        if(screen.width<=800)
        { //宽度小于等于 800 时, 不显示对联
            for(var i=0;i<collection.length;i++)
            {
                collection[i].object.style.display = 'none';
            }
            return;
        }
        for(var i=0;i<collection.length;i++)
        {
            var followObj = collection[i].object;
            var followObj_x = (typeof(collection[i].x)=='string'? eval
(collection[i].x):collection[i].x);
            var followObj_y = (typeof(collection[i].y)=='string'? eval
(collection[i].y):collection[i].y);
            if(followObj.offsetLeft!=(document.body.scrollLeft+followObj_x)) {
                var dx=(document.body.scrollLeft+followObj_x-followObj. offsetLeft) *delta;
                dx=(dx>0?1:-1)*Math.ceil(Math.abs(dx));
                followObj.style.left=followObj.offsetLeft+dx;
            }
        }
    }
}
```



```

        if(followObj.offsetTop!=(document.body.scrollTop+followObj_y)) {
            var dy=(document.body.scrollTop+followObj_y-followObj.offsetTop) *delta;
            dy=(dy>0?1:-1)*Math.ceil(Math.abs(dy));
            followObj.style.top=followObj.offsetTop+dy;
        }
        followObj.style.display = '';
    }
}

var theFloaters= new floaters(); //创建悬浮对联广告
//添加广告
theFloaters.addItem('div1','document.body.clientWidth-135',0,'</a><br> <a href=
"http://www.mimimama.com/clan.php?clanid=498" target="_blank"></a>');
theFloaters.addItem('div2',20,0,'<br><a href="http://www.mimimama.com/ clan.php?
clanid=498" target="_blank"></a>');
theFloaters.play(); //显示
</script>
</body>
</html>

```

【运行效果】

对联广告的运行效果如图 15-17 所示。

【难点剖析】

本例的重点是如何动态往页面中添加 div，并在指定的位置显示。动态添加元素本例使用的是“document.write”方法，是最常用也是最简单的输出文本流的方法。关于位置的选取请参考本例的“Play”方法，其中使用了“eval”方法来实现字符串格式到数值格式的转换。

15.22 带关闭的对联广告

【实例描述】

大型综合网站一般都提供对联广告，但为了不影响用户操作，允许用户关闭这些广告。本例提供一个可关闭的对联广告效果。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>

<DIV id=ad_d101

```

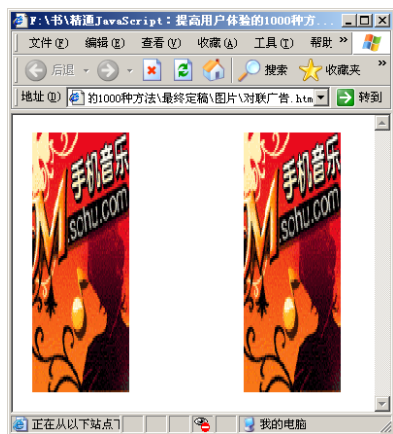


图 15-17 对联广告的运行效果



```
style="Z-INDEX: 1; LEFT: 5px; VISIBILITY: visible; WIDTH: 100px; POSITION: absolute; TOP: 55px">
<TABLE cellSpacing=0 cellPadding=0 width=100 border=0>
  <TBODY>
    <TR>
      <TD align=left><A onClick="ad_d101.style.visibility='hidden'">关闭</A> </TD></TR>
    <TR>
      <TD>
    </TD></TR></TBODY></TABLE></DIV>

<DIV id=ad_d102
style="Z-INDEX: 1; RIGHT: 5px; VISIBILITY: visible; WIDTH: 100px; POSITION: absolute; TOP: 55px">
<TABLE cellSpacing=0 cellPadding=0 width=100 border=0>
  <TBODY>
    <TR>
      <TD align=left><A onClick="ad_d102.style.visibility='hidden'">关闭</A></TD></TR>
    <TR>
      <TD>
    </TD></TR></TBODY></TABLE></DIV>
</body>
</html>
```



图 15-18 本例的运行效果

【运行效果】

本例的运行效果如图 15-18 所示。

【难点剖析】

本例的重点是页面的布局。两个对联广告使用两个 div 层构成，在层中使用了表格，其中实现“关闭”功能的是一个链接。关闭功能其实是将 div 的样式属性“visibility”设置为“hidden”，表示 div 不可见。

15.23 到边界反弹的漂浮图片

【实例描述】

很多网站的图片广告是以漂浮的形式展现，当图片漂浮到边界时会被弹回，然后继续向反方向移动。本例学习如何实现图片的反弹。

【实现代码】

```
<SCRIPT language=JavaScript>
var brOK=false;
var mie=false;
var aver=parseInt(navigator.appVersion.substring(0,1));
```

```

var aname=navigator.appName;
//检测浏览器类型
function checkbrOK()
{
    if(aname.indexOf("Internet Explorer")!=-1)
    {if(aver>=4) brOK=navigator.javaEnabled();
      mie=true;
    }
    if(aname.indexOf("Netscape")!=-1)
    {if(aver>=4) brOK=navigator.javaEnabled();}
}
var vmin=2;
var vmax=5;
var vr=2;
var timer1;
function Chip(chipname,width,height)
{this.named=chipname;
  this.vx=vmin+vmax*Math.random();
  this.vy=vmin+vmax*Math.random();
  this.w=width;
  this.h=height;
  this.xx=0;
  this.yy=0;
  this.timer1=null;
}
//移动图片
function movechip(chipname)
{
    if(brOK)
    {eval("chip="+chipname);
      if(!mie)
      {pageX=window.pageXOffset;
        pageW=window.innerWidth;
        pageY=window.pageYOffset;
        pageH=window.innerHeight;
      }
    }
    else
    {pageX=window.document.body.scrollLeft;
      pageW=window.document.body.offsetWidth-8;
      pageY=window.document.body.scrollTop;
      pageH=window.document.body.offsetHeight;
    }
    chip.xx=chip.xx+chip.vx;
    chip.yy=chip.yy+chip.vy;
    chip.vx+=vr*(Math.random()-0.5);
    chip.vy+=vr*(Math.random()-0.5);
    if(chip.vx>(vmax+vmin)) chip.vx=(vmax+vmin)*2-chip.vx;
    if(chip.vx<(-vmax-vmin)) chip.vx=(-vmax-vmin)*2-chip.vx;
}

```



```

if(chip.vy>(vmax+vmin)) chip.vy=(vmax+vmin)*2-chip.vy;
if(chip.vy<(-vmax-vmin)) chip.vy=(-vmax-vmin)*2-chip.vy;
if(chip.xx<=pageX) //水平方向不到边界时
{
    chip.xx=pageX;
    chip.vx=vmin+vmax*Math.random();
}
if(chip.xx>=pageX+pageW-chip.w) //水平方向超过边界时
{
    chip.xx=pageX+pageW-chip.w;
    chip.vx=-vmin-vmax*Math.random();
}
if(chip.yy<=pageY) //垂直方向不到边界时
{
    chip.yy=pageY;
    chip.vy=vmin+vmax*Math.random();
}
if(chip.yy>=pageY+pageH-chip.h) //垂直方向超过边界时
{
    chip.yy=pageY+pageH-chip.h;
    chip.vy=-vmin-vmax*Math.random();
}
if(!mie)
{
    eval('document.'+chip.named+'.top='+chip.yy);
    eval('document.'+chip.named+'.left='+chip.xx);
}
else
{
    eval('document.all.'+chip.named+'.style.pixelLeft='+chip.xx);
    eval('document.all.'+chip.named+'.style.pixelTop='+chip.yy);
}
chip.timer1=setTimeout("movechip('"+chip.named+"')",100); //设置定时器
}
}
//终止图片的循环漂浮
function stopme(chipname)
{
    if(brOK)
    {
        eval("chip="+chipname);
        if(chip.timer1!=null)
        {clearTimeout(chip.timer1)}
    }
}
var moveimg;
function beginmove()
{
    checkbrOK();
    moveimg=new Chip("moveimg",120,150);
    if(brOK)
    { movechip("moveimg");}
}
</SCRIPT>

```

需要在 body 中添加一个 ID 为“moveimg”的 div，其中包含要漂浮的图片，代码如下所示。

```
<body onload="beginmove()">
<p> <DIV id="moveimg" style="POSITION: absolute">
</p>
</body>
```

【运行效果】

图片的漂浮效果如图 15-19 所示。

【难点剖析】

本例的难点是随机移动的像素值和边界值的判断。在设置随机数时，本例使用了“Math.random()”，其中“Math”是 JavaScript 的常用计算对象，“random”用来生成一个随机数。页面边界值的获取是通过 body 的相关属性，如“scrollLeft”是获取最左侧的坐标。



图 15-19 图片的漂浮效果

15.24 用键盘控制图片移动

【实例描述】

本例学习如何实现可用键盘控制图片的上、下、左、右移动，类似在游戏中的效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<script language="JavaScript">
var key=0
var timer
function setObj()
{
    ietype = (document.layers) ? 1 : 0; //判断浏览器类型
    divObj = (ietype)? document.mydiv : mydiv.style //获取指定的div
    Xpos = parseInt(divObj.left); //获取div的x坐标
    Ypos = parseInt(divObj.top); //获取div的y坐标
    document.onkeydown = keyDown; //设置按键事件
    document.onkeyup = keyUp; //设置键盘弹起的事件
    if (ietype) document.captureEvents(Event.keydown | Event.keyup);
}
function keyDown(e) //按键的操作
{
    key = (ietype)? e.which : event.keyCode
    //判断用户按下的键，注意此键盘包括方向键和小键盘（数字键）
    if (key == 108 || key == 37) moveObj(1,2);
```



```
    if (key == 114 || key == 39) moveObj(1,3);
    if (key == 100 || key == 40) moveObj(1,4);
    if (key == 117 || key == 38) moveObj(1,5);
}
function keyUp(e)                                //按键弹起的操作
{
    key=0;clearTimeout(timer);
}
function moveObj(t,u)                            //移动图片的方法
{
    clearTimeout(timer)
    if (t==1){
        //根据移动的键，改变div的x和y坐标
        if (u==2){divObj.left = Xpos-=5;timer = setTimeout("moveObj(1,2)", 40);}
        if (u==3){divObj.left = Xpos+=5;timer = setTimeout("moveObj(1,3)", 40);}
        if (u==4){divObj.top = Ypos+=5;timer = setTimeout("moveObj(1,4)", 40);}
        if (u==5){divObj.top = Ypos-=5;timer = setTimeout("moveObj(1,5)", 40);}
    }
}
</script>
<body OnLoad="setObj();focus()">
<div id="mydiv" style="position:absolute; left:0px; top:80px;">
    
</div></body>
</html>
```

【难点剖析】

本例的重点是判断键盘的按键，以及动态改变图片的位置。图片所在的位置是一个 div 层，首先通过控件的“Left”和“Top”属性，获取此层的 X 坐标和 Y 坐标。然后通过“event.keyCode”属性获取用户的按键。最后根据按键在“moveObj”方法中改变 div 控件的坐标。

15.25 预装载图片提高站点速度

【实例描述】

页面通常会提供很多图片，为了提高速度可以预装载图片，当用户选择图片的时候，可从缓存中直接提取图片，从而达到提高速度的目的。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
    <script language="javascript">
//用数组存储需要的图片
var imgList = [];
imgList[0] = "logol.gif" ;
```

```

imgList[1] = "logo2.gif" ;
//用户单击按钮时，提取图片的方法
function getImage(index)
{
    var imgsrc= imgList[index];
    document.getElementById("img1").src=imgsrc;
}
</script>
</head>
<body>
<table id="mytbl" width="300" height="50" border="0" cellspacing="2" cellpadding="0"
bgcolor="#FFb609">
<tr>
<td> 显示第一张图片<input id="Button1" type="button" value="显示" onclick= "getImage(0)"
/></td><td><img id="img1" src=""></td>
</tr>
<tr>
<td> 显示第二张图片<input id="Button1" type="button" value="显示" onclick= "getImage(1)"
/></td><td>预先装载图片</td>
</tr>
</table>
</body>
</html>

```

【运行效果】

装载图片效果如图 15-20 所示。

【难点剖析】

本例的重点是如何加载图片并保存在页面中。“imgList”是一个全局数组，其中保存了当前需要的图片，单击按钮时根据指定的参数获取对应的图片。

15.26 始终在屏幕右下角的图片

【实例描述】

不管页面怎么滚动，当前图片都显示在窗体的右下角。本例学习如何保持这种移动状态。

【实现代码】

```

<SCRIPT LANGUAGE="JavaScript">
function setVariables() {
imgwidth=235; //图片的宽度
imgheight=19; //图片的高度
if (navigator.appName == "Netscape") { //Netscape 下的位置设置
    horz=".left";
    vert=".top";
    docStyle="document.";

```



图 15-20 装载图片效果



```
styleDoc="";
innerW="window.innerWidth";
innerH="window.innerHeight";
offsetX="window.pageXOffset";
offsetY="window.pageYOffset";
}
else { //IE 下的位置设置
    horz=".pixelLeft";
    vert=".pixelTop";
    docStyle="";
    styleDoc=".style";
    innerW="document.body.clientWidth";
    innerH="document.body.clientHeight";
    offsetX="document.body.scrollLeft";
    offsetY="document.body.scrollTop";
}
}
function checkLocation()
{
    objectXY="rightBottom"; //获取始终显示在右下角的 div
    var availableX=eval(innerW); //最大 x 坐标
    var availableY=eval(innerH); //最大 y 坐标
    var currentX=eval(offsetX); //鼠标当前的 x 坐标
    var currentY=eval(offsetY); //鼠标当前的 y 坐标
    x=availableX-(imgwidth+30)+currentX; //最终 div 的 x 坐标
    y=availableY-(imgheight+20)+currentY; //最终 div 的 y 坐标
    evalMove();
    setTimeout("checkLocation()",10); //定时移动
}
function evalMove() { //移动到指定位置
    eval(docStyle + objectXY + styleDoc + horz + "=" + x);
    eval(docStyle + objectXY + styleDoc + vert + "=" + y);
}
</script>
```



图 15-21 图片的移动效果

【运行效果】

图片的移动效果如图 15-21 所示。

【难点剖析】

本例的重点是使用“checkLocation”方法，用来设计当前 div 应该显示的位置。主要是根据鼠标的移动事件，获取 div 层的(X,Y)坐标，根据坐标不断地改变 div 的位置，让其永远显示在窗体的右下方。

15.27 可拖动的图片

【实例描述】

img 控件可以在网页中显示图片，但图片并不能拖动。本例学习如何实现图片的拖动效果。

【实现代码】

```

<style>
.drag{position:relative;cursor:hand}
</style>
<script language="JavaScript">
var dragapproved=false;
var z,x,y;
//移动图片
function move()
{
if (event.button==1&&dragapproved){
z.style.pixelLeft=temp1+event.clientX-x;
z.style.pixelTop=temp2+event.clientY-y;
return false;
}}
function drags()
{
if (!document.all)
return;
if (event.srcElement.className=="drag"){
dragapproved=true;
//以下设置拖动时的位置
z=event.srcElement;
temp1=z.style.pixelLeft;
temp2=z.style.pixelTop ;
x=event.clientX;
y=event.clientY;
document.onmousemove=move;
document.onmousedown=drags;
document.onmouseup=new Function("dragapproved=false");
}
}
//绑定鼠标移动事件
//绑定鼠标单击事件
//鼠标 up 事件
</script>

```

需要在 body 中添加一个 img 控件，代码如下所示：

```

```

【运行效果】

图片的拖动效果如图 15-22 所示。

【难点剖析】

本例中实现图片拖动使用了两个方法，一个用来拖放图片 (drags)，一个用来移动图片 (move)。拖放图片时让 img 图片的坐标跟随鼠标的坐标，移动图片就是实现图片移动到指定的位置。

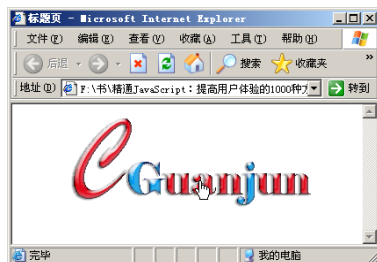


图 15-22 图片的拖动效果

15.28 等比例缩略图

【实例描述】

缩略图是相册网站和图片新闻网站常用的功能。本例学习如何实现等比例缩略图。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <script language="javascript">
function Wa_SetImgAutoSize()
{
var img=document.all.img1;           //获取图片
var MaxWidth=200;                     //设置图片宽度界限
var MaxHeight=100;                   //设置图片高度界限
var HeightWidth=img.offsetHeight/img.offsetWidth; //设置高宽比
var WidthHeight=img.offsetWidth/img.offsetHeight; //设置宽高比
if(img.readyState!="complete")return false; //确保图片完全加载
if(img.offsetWidth>MaxWidth){
img.width=MaxWidth;
img.height=MaxWidth*HeightWidth;
}
if(img.offsetHeight>MaxHeight){
img.height=MaxHeight;
img.width=MaxHeight*WidthHeight;
}
}
</script>
</head>
<body>


</body>
</html>
```



图 15-23 缩略图和原图的对比效果

【运行效果】

缩略图和原图的对比效果如图 15-23 所示。

【难点剖析】

本例首先设置了缩略图的最大高度和宽度，比例为 1:2。然后利用原图片的高度和宽度，获得图片的宽高比和高宽比。最后调整 img 控件的高度和宽度。

15.29 用 JavaScript 导出图片到 Excel

【实例描述】

用 JavaScript 可以轻松地将文字导入 Excel 中。本例将学习如何将图片也导入 Excel 中。

【实现代码】

```
<SCRIPT LANGUAGE="javascript">
function AutomateExcel()
{
var oExcel = new ActiveXObject("Excel.Application");      //创建 Excel 对象
var oWork = oExcel.Workbooks.Add();                      //新建一个 Excel 工作簿
var oSheet = oWork.ActiveSheet;                          //指定要写入内容的工作表为活动工作表
var table = document.all.myTbl;                          //指定要写入的数据源的 id
var myRow = table.rows.length;                           //取数据源行数
var myCell = table.rows(0).cells.length;                 //取数据源列数

for (i=0;i<myRow;i++){//在 Excel 中写行
    for (j=0;j<myCell;j++){//在 Excel 中写列
        //定义格式
        oSheet.Cells(i+1,j+1).Font.Bold = true;          //加粗
        oSheet.Cells(i+1,j+1).Font.Size = 10;            //字体大小
        if(table.rows(i).cells(j).innerHTML.toLowerCase().indexOf('<img')!=-1){//如果其 HTML 代码包括<img
            oSheet.Cells(i+1,j+1).Select();               //选中 Excel 中的单元格

oSheet.Pictures.Insert(table.rows(i).cells(j).getElementsByTagName('img')[0].src);
//插入图片

        }
        else{
            oSheet.Cells(i+1,j+1).value = table.rows(i).cells(j).innerText;    //向单元格写入值
        }
    }
}
oExcel.Visible = true;
oExcel.UserControl = true;
}
</SCRIPT>
```

【运行效果】

本例的运行效果如图 15-24 所示。图片导入 Excel 中的效果如图 15-25 所示。

【难点剖析】

本例的重点是 Excel 组件的运用。如果是普通文本，可以直接使用 Excel 单元格文本等于表格单元格文本的方法。但如果表格内是图片，则必须使用“Pictures.Insert”方法将图片插入到 Excel 中。



图 15-24 本例的运行效果



图 15-25 图片导出到 Excel 中的效果

15.30 使用 VML 打造可改变大小的圆框

【实例描述】

在网页中画一个图片并不是简单的事情，因为其不提供 C#语言中的 GDI 类库。本例学习如何使用 VML 轻松地画一个圆框。VML 相当于 IE 里的画笔，能实现几乎所有想要的图片，而且结合 JavaScript 脚本还可以让图片产生动态的效果。

【实现代码】

```
<html Xmlns:v="urn:schemas-microsoft-com:vml">
<style>
<!--
v\:* {behavior:url(#default#VML);}
-->
</style>
<body onmousemove='with(document.all.test.style){width=event.x;height= event.y;}'>
<v:oval id=test style="width: 100; height: 100">
<v:stroke weight="1px" color="navy"/>
</v:oval>
</body>
</html>
```



图 15-26 本例的运行效果

【运行效果】

本例的运行效果如图 15-26 所示。

【难点剖析】

在 VML 里标记使用的是 XML 扩展,这需要一个命名空间,可以使用惯用的“v”作为命名空间,这就形成本例代码的第一行。样式表中的内容主要作用是把命名空间“v”和系统预定义的 VML 行为连接起来。“v:oval”和“v:stroke”是 VML 中的一些图片分类。具体的图片定义可参考 VML 的相关资料。

15.31 JavaScript 实现文档结构图

【实例描述】

组织结构图是 Word 中经常遇到的图片结构。本例将学习如何利用 JavaScript 代码实现一个类似的组织结构图。

【实现代码】

```
<script language="javascript">
var dItem = new Array();
var w = 600;
var h = 40;
var iw = 60;
var ih = 70;
var boxh = 70;
var startleft = 350;
var starttop = 30;
var hr = "<hr size=\"1\" noshade>"
var labledv = "<div class=\"ItemCss\" style=\""
var hdv = "<div class=\"divhline\" style=\"width:\";
var vdv = "<div class=\"divvline\" style=\"height:\" + h + \"px;\"";
var endsdv = "\">";
var enddv = "</div>";
var htm = "";
var len;
var maxn=0;                                     //深度初始变量

function createStruct()
{
    dItem[0] = "1|教育部|0|";                    //设置要显示的结构内容，以“|”间隔
    dItem[1] = "2|东部教育局|1|";                //注意第三项为父级节点
    dItem[2] = "3|西部教育局|1|";
    dItem[3] = "4|东部高校|2|";
    dItem[4] = "5|东部高中|2|";
    dItem[5] = "6|商务代表|5|";
    dItem[6] = "7|东部初中|2|";
    dItem[7] = "8|西部高校|3|";
    dItem[8] = "9|西部高中|3|";
    len = dItem.length;                          //获取结构数组的长度
    Set_Item(0,0);                                //设置开始的节点
    Set_Max();
    Write_Item(0,0,0,1);
    var html = "";
    for(var i=0;i<len;i++)
    {
        html = html +dItem[i]+"<br>";
    }
}
```



```

document.getElementById("divStruct").innerHTML = htm;
}

function Set_Item(pid,ni)                                     //设置节点层次
{
    var n = ni + 1;                                          //子节点的序号
    var iAry = new Array();
    for(var i=0;i<len;i++)
    {
        iAry = dItem[i].split("|");                        //分解数组中的当前节点
        if(iAry[2] == pid)                                   //是当前节点的子节点
        {
            dItem[i] = dItem[i] + ni;                       //当前节点中添加一项
            if(maxn < ni)                                     //改变结构的深度
            {
                maxn = ni;
            }
            Set_Item(iAry[0],n);                             //循环设置层次
        }
    }
}

function Set_Max()                                           //设置节点子节点中最大数
{
    var iAry = new Array();
    var childnum;
    for(var i=0;i<len;i++)
    {
        iAry = dItem[i].split("|");                        //分解数组中的当前节点
        childnum = Get_Child_Num(iAry[0]);                 //获取当前节点的子节点
        if(childnum <= 1)                                   //如果子节点数为 0 或 1
        {
            dItem[i] = dItem[i] + "|0";                   //当前节点中添加一项
        }
        else
        {
            dItem[i] = dItem[i] + "|" + Get_Max(iAry[0],iAry[3]); //获取最大节点
        }
    }
}

function Get_Max(pid,start)                                  //获取指定节点的最大节点
{
    var iAry = new Array();
    var m = 0;
    var n = 0;
    for(var j=start;j<=maxn;j++)
    {
        for(var i=0;i<len;i++)

```

```

{
    iAry = dItem[i].split("|");
    if(iAry[3] == j)
    {
        if(Get_RootID(pid,iAry[0]))                //获取当前节点的根节点
        {
            m = m + 1;
        }
    }
    if(n < m)
    {
        n = m;
    }
}
m = 0;
}
return n;
}
function Get_RootID(pid,id)                        //获取当前节点的根节点
{
    var iAry = new Array();
    for(var i=0;i<len;i++)
    {
        iAry = dItem[i].split("|");                //分解当前节点
        if(iAry[0] == id)
        {
            if(iAry[2] == pid)
            {
                return true;
                break;
            }
            else
            {
                return Get_RootID(pid,iAry[2]);      //返回根节点
            }
        }
    }
    return false;
}

function Get_Item(id)                              //取得指定节点号所在的数组
{
    var i;
    var items;
    var iAry = new Array();
    for(i=0;i<len;i++)                             //边界节点组
    {
        iAry = dItem[i].split("|");                //分解当前节点

```



```

    if(iAry[0] == id)
    {
        items = dItem[i];           //获取节点
        break;
    }
}
return items;
}

function Get_Child_Num(pid)        //根据父节点取得子节点个数
{
    var i;
    var rnum = 0;
    var iAry = new Array();
    for(i=0;i<len;i++)             //遍历组织结构数组
    {
        iAry = dItem[i].split("|"); //将每一项再分离出数据
        if(iAry[2] == pid)         //第三项便是父节点
        {
            rnum = rnum + 1;       //是当前节点的子节点
        }
    }
    return rnum;
}

function Write_Item(ipid,ltmp,wtmp,cnt)
{
    var iAry = new Array();
    var id;
    var txt;
    var pid;
    var lens;
    var maxnum;
    var t;
    var l;
    var hline_width;
    var dvline = "";
    var childnum = 0;
    var itxt;
    var tmpcnt = 0;
    for(var i=0;i<len;i++)
    {
        itxt = dItem[i];
        iAry = itxt.split("|"); //分解节点项
        if(iAry[2] == ipid)
        {
            id = iAry[0];
            txt = "<a href=?id=" + id + "\">" + iAry[1] + "</a>";

```


//动态创建链接, 链接内容来自节点数据

```

pid = iAry[2];
lens = iAry[3];
maxnum = iAry[4];
childnum = Get_Child_Num(id);
hline_width = maxnum * iw;
if(pid == 0)
{
    t = starttop;
    l = startleft;
}
else
{
    t = starttop + 2 * lens * h + lens * ih;
    l = ltmp - wtmp/2 + (wtmp / 2) * tmpcnt;
}
dvline = "";
if(childnum > 1)
{
    var t1;
    var l1;
    var t2;
    var l2;
    var w2;
    t1 = t + ih;
    l1 = l + l2;
    w2 = hline_width/2;
    t2 = t1 + h;
    l2 = l - w2 + 10;    //使用div 实现边框效果
    dvline = "<div class=\"divvline\" style=\"height:\" + h + \"px;left:\" + l1 + \"px;top:\"
+ t1 + \"></div>\";
    dvline = dvline + "<div class=\"divhline\" style=\"width:\" + hline_width + \"px;left:\"
+ l2 + \"px;top:\" + t2 + \"></div>\";
    for(var j=0;j<childnum;j++)
    {
        var t3;
        var l3;
        t3 = t1 + h;
        l3 = l2 + (hline_width/(childnum-1)) * j;
        var tmpline = "<div class=\"divvline\" style=\"height:\" + h + \"px;left:\" + l3
+ \"px;top:\" + t3 + \"></div>\";
        dvline = dvline + tmpline;
    }
    dvline = dvline
}
else if(childnum == 1)
{

```



```
var t4;  
var l4;  
l4 = 1 + 12;//使用 div 实现边框效果  
dvline = "<div class=\"divvline\" style=\"height:\" + h + \"px;left:\" + l4 + \"px;top:\" +  
(t + ih) + \"></div>\";  
dvline = dvline + "<div class=\"divline\" style=\"height:\" + h + \"px;left:\" + l4 + \"px;top:\"  
+ (t + ih + h) + \"></div>\";  
  
}  
htm = htm + labledv + "left:\" + l + \"px;top:\" + t + \"px\" + endsdv + txt + enddv + dvline;  
if(cnt % 2 == 0)  
{  
    tmpcnt = tmpcnt + 2;  
}  
else  
{  
    tmpcnt = tmpcnt + 1;  
}  
Write_Item(id,l,hline_width,childnum);  
//循环输出  
  
}  
}  
}  
</script>
```

【运行效果】

组织结构图的效果如图 15-27 所示。

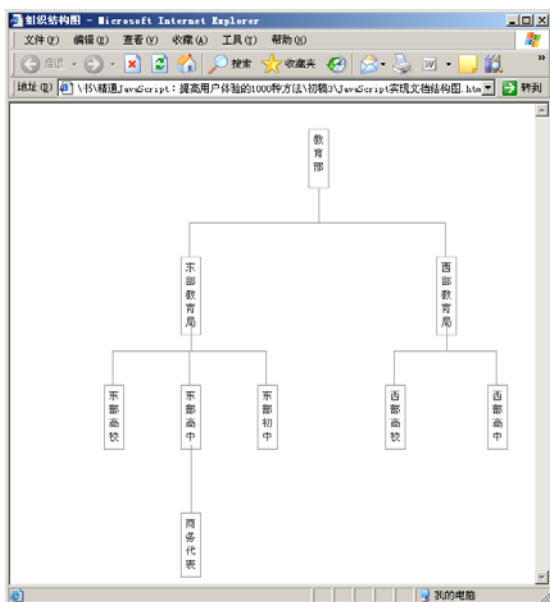


图 15-27 组织结构图的效果

【难点剖析】

组织结构图的难点是如何实现一种组织结构形式。本例中将这此结构数据保存在数组中，同时设置数组中的每项又包含 3 个数据：当前节点序号、节点文本和当前节点的父节点。在数据处理时，使用字符串对象的“split”方法分解这些数据，并进行节点判断。

15.32 判断一幅图片是否加载完毕

【实例描述】

有时候需要在图片加载完成后调整界面中各个元素的位置。本例介绍如何判断图片是否加载完毕。

【实现代码】

```
<html>
<head>
<title>Untitled</title>
</head>
<body>
<img border=0 src=logol.gif onload="alert('加载完毕')">
</body>
</html>
```

【运行效果】

实例的运行效果如图 15-28 所示。

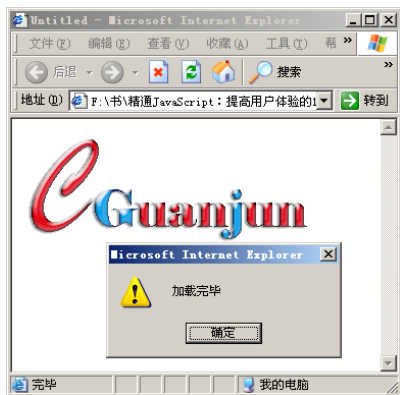


图 15-28 实例的运行效果

【难点剖析】

本例重点使用了“onload”事件，此事件在图片加载完毕后被触发。可在“onload”事件中添加一个方法用来判断图片的位置、高度和宽度等，实现页面的调整。

第 16 章

页面数据的验证

本章导读

网页可以被世界各地的人访问，所以网页的安全性就是一个很大的问题。本章通过一些小技巧学习如何保护自己的网页，包括验证用户的输入、制作网站验证码等。

16.1 验证字符串是否全由数字组成

【实例描述】

验证字符串通常使用正则表达式完成。本例通过一个简单正则表达式的应用学习如何验证字符串是否全部由数字组成。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function checkNum(str)
{

    if( str.match(/\d/)== null)
        alert("输入数值错误! ");
    else
        alert("数值正确! ");
}
</script>
</head>
<body>
<input type="text" name="txt1" value="a">
<input type="button" name="btn1" value="测试" onclick="checkNum(txt1.value)">
</body>
</html>
```

【难点剖析】

本例的重点是正则表达式的用法，判断数字的正则表达式符号是“\d”。“Match”方法使用正则表达式模式对字符串进行查找，并将查找结果作为数组返回。具体正则表达式符号的应用可参考介绍正则表达式的专业书籍。

16.2 验证表单项必须填写

【实例描述】

在注册用户的时候，用户名和密码必须填写。本例学习如何实现对比必填项的判断。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function CheckForm()
```



```
{
//判断输入框内容的长度
if ( document.getElementById("txtname").value.length == 0) {
    alert("姓名必须输入!");           //提示用户
    document.getElementById("txtname").focus();       //文本框获取焦点
    return false;
}
return true;
}
</script>
</head>
<body>
请输入姓名: <input type=text name="txtname" value="" onblur="CheckForm()">
</body>
</html>
```

【难点剖析】

本例的难点在于如何判断用户未填写任何内容。代码中使用“length”来判断输入框内容的长度，如果长度为“0”则表示未填写内容。

16.3 判断用户输入是否为中文

【实例描述】

判断用户输入的文本类型可以使用正则表达式实现，也可以根据文本的 unicode 编码实现。本例学习如何使用 unicode 编码实现对中文字符的判断。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function ischinese(s){
    var ret=true;
    for(var i=0;i<s.length;i++)           //遍历每一个文本字符
        ret=ret && (s.charCodeAt(i)>=10000);       //判断文本字符的 unicode 值
    return ret;
}
function chk(str)
{
    var val= ischinese(str);               //判断是否包含中文
    if(!val)
        alert("您输入的字符不是全中文");
    else
        alert("您输入的字符是全中文");
}
```

```

</script>
</head>
<body>
<input type="text" name="txt1" value="this is test!">
<input type="button" value="转换文本" onClick="javascript:chk(txt1.value)">
</body>
</html>

```

【运行效果】

本例的运行效果如图 16-1 所示。

【难点剖析】

本例的重点是字符的 unicode 编码。字符串对象提供一个“charCodeAt”方法，专门用来获取字符的 unicode 编码，中文编码必定大于“10000”。

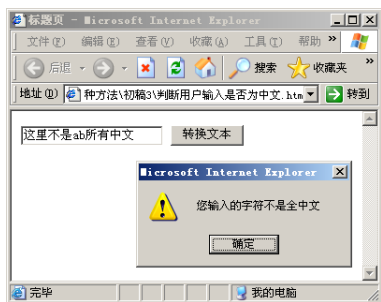


图 16-1 本例的运行效果

16.4 验证列表框中的值是否重复

为了方便用户填写资料，可以下拉框的形式让用户选择值代替直接填写。但有时候下拉框并不能提供全部数据，此时需要用户自己填写。本例用来判断用户填写的数据是否已经存在于下拉框中。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function chk()
{
    var txtValue = document.getElementById("txtValue"); // 获取文本框的值
    var sel = document.getElementById("sel"); // 找到下拉框
    for (var i=0; i<sel.length; i++) // 遍历下拉框中的所有数据
    {
        if (txtValue.value==sel.options[i].text){ // 如果输入的值等于下拉框中的值
            alert("该值已经存在");
        }
    }
}
</script>
</head>
<body>
<div id="mydiv">
<select id="sel" name="sel">
<option>China</option>
<option>England</option>

```

```
<option>German</option>
</select>
<input type="text" name="txtValue" id="txtValue"><br/>
<input type="button" value="检测是否存在" onclick="chk();">
</div>
</body>
</html>
```

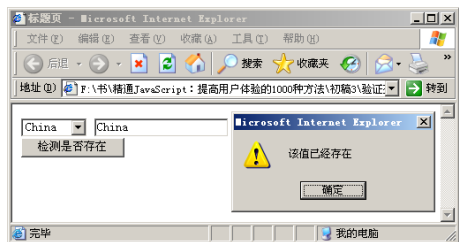


图 16-2 本例的运行效果

【运行效果】

本例的运行效果如图 16-2 所示。

【难点剖析】

本例首先获取文本框的值，然后对下拉框中的值进行遍历，这使用的是“for”循环语句。“sel.options[i].text”用来获取下拉框中的当前值，用此值比较用户输入的值，如果相等，则表示用户输入的数据重复。

16.5 检测输入框的统一方法

【实例描述】

在一些资料填写页面，因为有些数据要求用户必须填写，所以通常会逐个判断输入框中是否有内容。本例将提供一个通用的方法，可以判断页面上所有的输入框中是否有内容。

【实现代码】

```
<html>
<head>
<title>检测输入框的统一方法</title>
<script language="JavaScript">
function chk()
{
    var b=document.getElementsByTagName("input")           //获取所有的输入框
    for(var i=0;i<b.length;i++){                             //遍历所有的输入框
        if(b[i].name.substr(0,4)=="item"){                   //判断输入框的前 4 位
            if(b[i].value==""){                               //如果值为空，给出提示
                alert("请输入第"+(i+1)+"项的值。");           //值空的文本框获得焦点
                b[i].focus();
                return false;                                  //返回
            }
        }
    }
    return true
}
</script>
</head>
<body>
```



```

<form name="form1" action="http://www.google.com" onSubmit="return chk()">
1.<input name="item1" type="text" value="" >
2.<input name="item2" type="text" value=""><br />
3.<input name="item3" type="text" value="">
4.<input name="item4" type="text" value=""><br />
<input name="btn" type="submit" value="提交">
</form>
</body>
</html>

```

【运行效果】

本例的运行效果如图 16-3 所示。

【难点剖析】

“getElementsByTagName(标签)”方法用来获取指定标签名称的控件，因为输入框是使用 input 控件，所以本例使用此方法获取页面中所有的 input 控件，这些控件也包含了具有 input 标签的提交按钮。通过 input 的“name”属性，判断此控件是否是输入框，如果是再继续判断是否填写了内容。

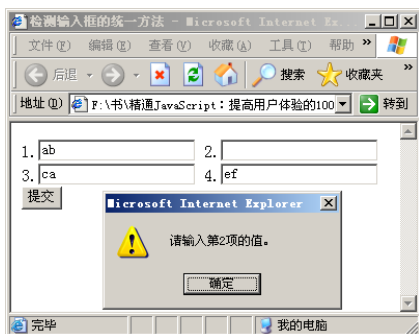


图 16-3 本例的运行效果

16.6 E-mail 的验证

【实例描述】

在网站中注册用户时，常通过 E-mail 来发送用户的注册密码。为了验证用户输入的 E-mail 是否正确，本例学习使用正则表达式验证用户输入的 E-mail 是否规范。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
    function ismail(mail)
    {
        //测试 E-mail 的正则表达式
        return(new RegExp(/^(\w+)((-\w+)|(\.\w+))*@[A-Za-z0-9]+((\(|-)[A-Za-z0-9]+)*\.
[A-Za-z0-9]+$/).test(mail));
    }
</script>
</head>
<body>
<input type="text" name="txt1" value="booknew@sina.com">
<input type="button" value="测试" onClick="javascript:alert(ismail(txt1.value))">
</body>
</html>

```



【难点剖析】

本例的重点是正则表达式在 JavaScript 中的应用。“RegExp”用来创建正则表达式，“test”方法用来验证用户输入的数据。其中正则符号的使用请参考详细的正则手册。

16.7 不使用正则验证 IP 地址

【实例描述】

IP 地址是一组有规律的字符串，可以使用正则来验证，也可以利用长度和大小来判断。本例利用长度和大小实现 IP 地址的验证。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
//测试 IP 的方法
function isip(s){
    var check=function(myValue){
        try{return (myValue<=255 && myValue>=0)}           //判断 IP 值的范围
        catch(x){return false}
    };
    var re=s.split(".");           //根据.将 IP 分隔为数组
    return (re.length==4)?(check(re[0]) && check(re[1]) && check(re[2]) &&
check(re[3])):false
}

var s="202.168.54.215";           //要测试的 IP
alert(isip(s));                   //显示对上述 IP 的测试
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是 IP 地址的书写规范。IP 地址共有 4 组数字，以“.”间隔，每组数字的大小不能超过 255。本例首先判断 IP 地址是否为 4 组数字，然后使用自定义的“check”方法判断值的大小是否超过 255。

16.8 IP 地址输入框

【实例描述】

IP 地址是一段很有规律的字符串，每三位用一个小数点隔开。本例通过此规律设置一个简

便的 IP 地址输入框。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style>
div.IPDiv{background:#ffffff;width:120;font-size:9pt;text-align:center;border:2 ridge
threeDShadow;border-right:inset threeDHighlight;border-bottom: inset threeDHighlight;
}
input.IPInput{width:24;font-size:9pt;text-align:center;border-width:0;
}
</style>
<script language="javascript" for="document" event="onkeydown">
alert(event.keyCode);
if(event.keyCode==13)                                //如果用户按下 Enter 键
event.keyCode=9;                                     //转换为 tab 键
</script>
<script language="javascript">
var tmpStr=[];                                       //创建一个数组
for(var i=0;i<4;i++)                                //通过循环实现每隔 3 位，输出一个点
tmpStr[i]="<input class=IPInput name=IPInput type=text size=3 maxlength=3
onkeydown='if(event.srcElement.value.length==3||event.keyCode==39)event.keyCode=9'>"+(i==
3?" ":".");
document.write("<div class=IPDiv>"+tmpStr.join("")+"</div>");           //输出字符串
</script>
</head>
<body>
</body>
</html>
```

【运行效果】

IP 地址输入框的效果如图 16-4 所示。

【难点剖析】

本例的重点是通过一个循环语句输出 4 个标准的 input 文本控件。注意这 4 个文本控件没有边框，用一个 div 封装。同时还设置了这些文本框的“maxlength”属性，保证每个框内只能输入三个字符。

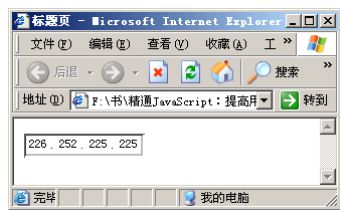


图 16-4 IP 地址输入框的效果

16.9 判断变量是否已经定义

【实例描述】

在普通的变量值判断中，可以使用“变量==null”来判断变量是否有值。该如何判断变量是否已经被定义？本例将学习这种判断方法。



【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function judge()
{
    if(typeof iTxt=="undefined")
        alert("iTxt 变量未定义")
    else
        alert("iTxt 变量已经定义");
}
</script>
</head>
<body>
    <input id="Button1" type="button" value="判断" onclick="judge()" />
</body>
</html>
```

【难点剖析】

本例的重点是“undefined”，用其判断某个变量是否被定义。如果已经在前面使用“var”定义了变量，则 undefined 不成立，否则认为变量未定义。

16.10 判断方法是否已经定义

【实例描述】

为了避免方法运行过程中出现错误，可在使用方法前先判断方法是否被定义。本例介绍如何判断方法是否已经被定义。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
if(typeof a!='function'){
    alert("方法 a 没有定义!");
}
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是“typeof”方法。“typeof”方法用来判断 JavaScript 中各种对象的类型，如控

件、方法和数据类型等。如果返回“function”则说明此对象是一个方法。

16.11 表单验证样式

【实例描述】

实现表单验证的样式有很多，本例将使用一种类似 Ajax 效果的验证方法实现一个分数（数值型）的验证。

【实现代码】

```
<script type="text/javascript">
var oScore = document.getElementById("txtScore");           //获取输入文本框
var oNote = document.getElementById("divTipInfo");           //获取提示信息框
var oAlarm = document.getElementById("divAlarmInfo");         //获取警告信息框
oScore.onfocus = function()
{
    oNote.className = "notetrue";                             //获取焦点时改变样式
}
oScore.onblur = function()                                     //失去焦点时的方法
{
    if (this.value != "")
    {
        var score = parseInt(this.value);                     //转换数值类型
        //判断输入内容是否符合条件
        if (isNaN(score) || !(score>=0 && score<=100))
        {
            oAlarm.style.display = "block";
            oNote.style.display = "none";
        }
        else
        {
            this.value = score;
            oAlarm.style.display = "none";
            oNote.className = "note";
            oNote.style.display = "block";
        }
    }
    else
    {
        //用户没有输入的情况下
        oAlarm.style.display = "none";
        oNote.className = "note";
        oNote.style.display = "block";
    }
}
```

【运行效果】

表单验证的效果如图 16-5 所示。

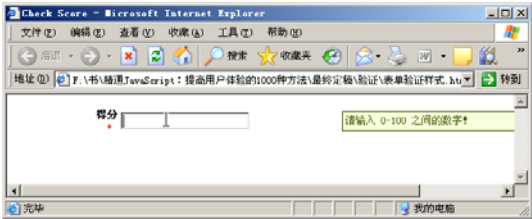


图 16-5 表单验证的效果

【难点剖析】

本例的重点是输入框的两个事件：“onfocus”和“onblur”。当输入框获得焦点时，改变提示框的样式，如果用户输入错误则显示警告信息框，隐藏提示框。本例中 CSS 样式表起到了很大的作用，具体样式代码可参考随书光盘。

16.12 判断表单是否已修改

【实例描述】

有些网站允许用户修改表单（如修改用户资料），如果用户没有修改则不需要将表单提交给服务器，如果修改了，则需要与服务器进行数据交互。本例演示如何判断用户是否对表单进行了修改。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
  <form>
    <input name="txt1" value="10"><br />
    <input name="txt2" value="20"><br />
    <input type="checkbox" value="5" />
    <input type="button" id="btnSave" value="保存" />
  </form>
  <script type="text/javascript">
    //判断表单是否已经修改的方法
    function IsModified()
    {
      var result = false;                                //初始化返回值
      var colInput = document.getElementsByTagName("input"); //获取输入框控件
      for (var i=0; i<colInput.length; i++)                //逐个判断页面中的 input 控件
      {
        if (colInput[i].value != colInput[i].defaultValue) //判断输入的值是否等于初始值
        {
          result = true;                                //如果不相等，则返回true，表示已经修改
        }
      }
    }
  </script>
</body>
</html>
```

```

        colInput[i].style.backgroundColor = "#ff9000";    //改变被修改控件的背景色
    }
}
return result;
}
document.getElementById("btnSave").onclick = function ()    //重定义按钮的单击事件
{
    if (IsModified())
    {
        return window.confirm("表单已经修改, 是否继续保存? ");
    }
}
</script>
</body>
</html>

```

【运行效果】

判断表单是否被修改的提示效果如图 16-6 所示。

【难点剖析】

本例的重点是文本框的“defaultValue”属性。“defaultValue”属性是文本框的默认值, 可以通过文本框的“Value”属性获取改变后的值。通过比较这两个值可以判断文本是否发生变化。

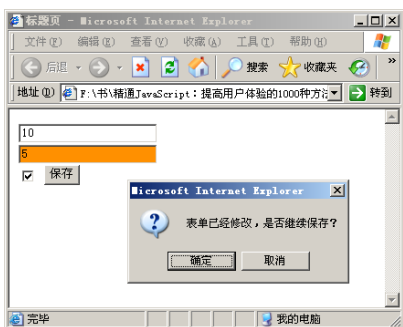


图 16-6 判断表单是否被修改的提示效果

16.13 判断控件的类型

【实例描述】

为了获取用户的操作(如选择单选还是复选), 有时候需要判断用户操作的控件类型。本例学习如何实现这种功能。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function test(frmObj)
{
    //考虑到 input 有多种类型, 所以还需要判断 type 属性
    if (frmObj.tagName=="INPUT")
        alert("您操作的控件类型是: "+ frmObj.type) ;
    else
        alert("您操作的控件类型是: "+ frmObj.tagName);
}
</script>

```



```
</head>
<body>
<input type=text name="txt1" value="this is test!" onchange="test(this)">
<input type=button value="判断" onClick="test(this)">
</body>
</html>
```

【难点剖析】

本例的重点是“tagName”属性和“type”属性。“tagName”返回控件在 HTML 中的标准名称，由于 input 控件有多种类型，所以还要单独判断。“type”用来返回这些控件的类型，如“button”、“text”等。

16.14 密码强度检查

【实例描述】

密码强度检查一般用在网站用户注册时，用来检查用户输入的密码是否安全，此功能也是 Ajax Web 2.0 技术的特色。本例学习使用 JavaScript 实现密码强度检查。

【实现代码】

```
<html>
<head>
<title>密码强度提示</title>
<style type="text/css">
#chkResult{margin-left:53px;height:15px;}
</style>
</head>
<body>
<form name="form1">
<label for="pwd">用户密码</label>
<input type="password" name="pwd" onblur="chkpwd(this)" />
<div id="chkResult"></div>
<label for="pwd2">重复密码</label>
<input type="password" name="pwd2" />
</form>
<script type="text/javascript">
//检测密码的方法
function chkpwd(obj)
{
var t=obj.value;
var id=getResult(t);
//定义对应的消息提示
var msg=new Array(4);
msg[0]="密码太短。";
msg[1]="密码强度比较差。";
msg[2]="密码强度一般。";
```



```

msg[3]="密码强度高。";
var sty=new Array(4);
sty[0]=-45;
sty[1]=-30;
sty[2]=-15;
sty[3]=0;
//消息提示对应的颜色
var col=new Array(4);
col[0]="gray";
col[1]="red";
col[2]="#ff6600";
col[3]="Green";

//设置显示效果
var sWidth=300;
var sHeight=15;
var Bobj=document.getElementById("chkResult");
Bobj.style.fontSize="12px";
Bobj.style.color=col[id];
Bobj.style.width=sWidth + "px";
Bobj.style.height=sHeight + "px";
Bobj.style.lineHeight=sHeight + "px";
Bobj.style.textIndent="20px";
Bobj.innerHTML="密码检测提示: " + msg[id];
}

//定义检测函数,返回 0/1/2/3 分别代表无效/差/一般/强
function getResult(s){
if(s.length < 4){
return 0;
}
var ls = 0;
if (s.match(/[a-z]/ig)){ ls++; }
if (s.match(/[0-9]/ig)){ ls++; }
if (s.match(/(.[^a-z0-9])/ig)){ ls++; }
if (s.length < 6 && ls > 0){ ls--; }
return ls
}
</script>
</body>
</html>

```

【运行效果】

密码强度提示效果如图 16-7 所示。

【难点剖析】

本例的难点在于判断用户输入的密码是字母还是数字，这里使用了正则表达式。“s.match(/[a-z]/ig)”用来检测字母，“s.match(/[0-9]/ig)”用来检测数字，“s.match(/(.[^a-z0-9])/ig)”

则检查密码是否既包含字母又包含数字。

16.15 身份证号的验证

【实例描述】

验证身份证号的主要条件是号码必须全部由数字组成且长度为 15 或 18 位。为了保证身份证号的正确性，还需要判断其中的年份和日期是否正确。本例通过一个正则表达式实现对身份证号的验证。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
    function isCardNo(num)
    {
        if (isNaN(num)) {alert("输入的不全是数字!"); return false;}
        var len = num.length, re;
        if (len == 15) //15 位身份证的判断
            re = new RegExp(/^(\d{6})()?\d{2}(\d{2})(\d{3})$/);
        else if (len == 18) //18 位身份证的判断
            re = new RegExp(/^(\d{6})()?\d{4}(\d{2})(\d{2})(\d{3})(\d)$/);
        else {alert("输入的数字位数不对!"); return false;}
        var a = num.match(re); //判断是否有符合条件的表达式
        if (a != null)
        {
            if (len==15) //对 15 位身份证中日期的判断
            {
                var D = new Date("19"+a[3]+"/"+a[4]+"/"+a[5]);
                var B = D.getYear()==a[3]&&(D.getMonth()+1)==a[4]&&D.getDate()==a[5];
            }
            else //对 18 位身份证中日期的判断
            {
                var D = new Date(a[3]+"/"+a[4]+"/"+a[5]);
                var B = D.getFullYear()==a[3]&&(D.getMonth()+1)==a[4]&&D.getDate()==a[5];
            }
            if (!B) {alert("输入的身份证号 "+ a[0] +" 里出生日期不对!"); return false;}
        }
        return true;
    }
</script>
</head>
<body>
<input type="text" name="txt1" value="1110120780423003">
<input type="button" value="判断身份证号" onClick="javascript:alert(isCardNo(txt1.value))">
```



图 16-7 密码强度提示效果

```
</body>
</html>
```

【难点剖析】

本例的重点是验证身份证号的条件。长度必须满足 15 或 18 位。在满足长度的条件后，再判断出生年月日是否符合日期规范。“match”方法判断是否能找到符合条件的匹配，没有匹配则返回“null”。

16.16 JavaScript 生成验证码（一）

【实例描述】

为了防止一些自动注册程序，在网站注册新用户或登录时通常需要填写验证码。本例学习如何使用 JavaScript 制作验证码。

【实现代码】

```
<HTML>
<HEAD>
<TITLE>生成验证码</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function createCode(len)
{
    var seed = new Array(
        'abcdefghijklmnopqrstuvwxyz',
        'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
        '0123456789'
    );
    //创建需要的数据数组

    var idx,i;
    var result = '';
    //返回的结果变量
    for (i=0; i<len; i++) //根据指定的长度
    {
        idx = Math.floor(Math.random()*3); //获得随机数据的整数部分-获取一个随机整数
        result += seed[idx].substr(Math.floor(Math.random()*(seed[idx].length)), 1);
        //根据随机数获取数据中一个值
    }
    return result;
    //返回随机结果
}
</SCRIPT>
</HEAD>
<BODY>
验证码长度:
<SELECT id="sel">
<option value=1>1</option>
<option value=3>3</option>
<option value=5>5</option>
<option value=7 selected>7</option>
```



```
<option value=9>9</option>
</SELECT>
<BR>
生成:
<INPUT TYPE="text" id="txtCode">
<INPUT TYPE="button" VALUE="生成" ONCLICK="txtCode.value=createCode(sel.value)">
</BODY>
</HTML>
```

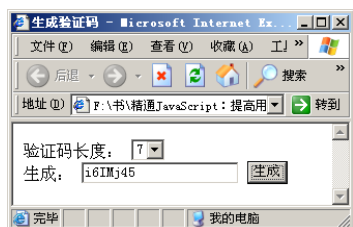


图 16-8 生成验证码的效果

【运行效果】

生成验证码的效果如图 16-8 所示。

【难点剖析】

本例的难点就是两个数学方法：“Math.random()”和“Math.floor()”。“Math.random()”产生 0~1 之间的随机数，而“Math.floor()”是获得一个数的整数部分，而不是四舍五入的整数。

16.17 JavaScript 生成验证码（二）

【实例描述】

很复杂的验证码会包含图片、字体、文字和数字的变化。本例学习一种比较简单的静态数字验证码。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
请输入验证码: <input type="text" name="codeTxt" size="4"><script>
var num=Math.floor(Math.random()*8999)+1000 //取 4 位随机数
function txtTest()
{
if(document.all.codeTxt.value!=num) //如果用户输入不正确
    alert("验证码错误!");
else
    alert("验证通过!");
return;
}
document.write(num) //显示随机验证码
</script>
<input type="button" value="确认" onclick="txtTest()"></body>
</html>
```

【运行效果】

生成验证码的效果如图 16-9 所示。

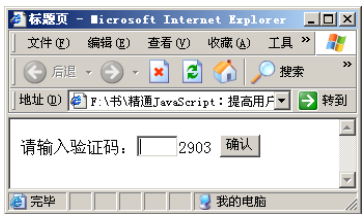


图 16-9 生成验证码的效果

【难点剖析】

本例的重点是验证码的生成。首先使用“Math.random()”方法生成一个 0 ~ 1 之间的随机数，然后用此结果乘以 8999 便得到三位数的随机数，再加 1000 便是标准的 4 位随机数。

第 17 章

进度条、滚动条的特效处理

本章导读

为了让用户的等待界面不至于太枯燥，在数据处理比较多的页面中通常提供一个进度条，以提示用户等待时间。本章介绍如何制作一些好看的进度条，同时还介绍如何设置页面中的各种滚动条。

17.1 使用符号制作的进度条

【实例描述】

进度条一般用来显示某个操作的执行进度，如下载文件、打开对话框等。本例学习制作一个简单的进度条。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<form name=loading>
  <p align=center> <font color="#0155cc" size="2" face="Arial">页面载入中，请稍等...
</font><br>
    <input type=text name=progress size=46 style="font-family:Arial; font-weight: bolder;
color:#0066ff; background-color:#fef4d9; padding:0px; border- style:none;">
    <br>
    <input type=text name=percent size=47 style="color:#0055BB; text-align: center;
border-width:medium; border-style:none;">
    <script language="javascript">
      var bar=0;                                //进度条的进度
      var line="||" ;                            //类似进度条的符号
      var amount="||" ;
      count() ;
      function count(){
        bar=bar+2 ;                                //进度条+2
        amount =amount + line ;                    //符号也跟着增加
        document.loading.progress.value=amount ;   //进度条显示符号
        document.loading.percent.value=bar+"%" ;   //显示进度
        if (bar<100)                                //判断进度条是否已经到头
          setTimeout( "count()",100);
        else
          window.location = "#";
      }
    </script>
  </p>
</form> </body>
</html>
```

【运行效果】

进度条的运行效果如图 17-1 所示。

【难点剖析】

本例的重点其实是进度条的布局。使用多个“||”符号构成，然后利用“amount= amount+line”



特效，仅使用两个简单的单元格完成。

语句，实现符号的不断增多，达到进度条的变化效果。此例可用于很普通的页面加载，但无法计算真正的加载时间。

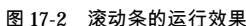
17.2 用 table 制作进度条

本例提供的进度条使用简单、方便，完全没有使用图片和

【运行效果】

【难点剖析】

本例的重点是如何获取进度，以及如何动态设置进度条的宽度。进度条的进度使用一个固定的常量“200”表示，每次进度显示使用



“i/2+”%”方式运算。动态设置单元格的宽度使用的是“setExpression”方法，其语法如下所示：

```
setExpression(attribute,value,language)
```

其中参数依次为属性、属性值、语言。

17.3 用 CSS+JS 制作进度条（一）

【实例描述】

本节的实例演示一个用 JavaScript 控制的进度条，其中使用 JavaScript 的一些标准语法，并引用 CSS 来设计进度条的样式。

【实现代码】

```
<script type="text/javascript">
    //设置 span 元素的编号
    var progressEnd = 16;
    //设置进度条的颜色为蓝色
    var progressColor = "blue";
    //设置进度条的走动时间——单位为毫秒
    var progressInterval = 350;
    //进度条的开始标志
    var progressBegin = 0;
    var progressTimer;
    function progress_clear()
    {
        //清空定时器
        clearTimeout(progressTimer);
        //隐藏 div
        document.getElementById("framediv").style.visibility="hidden";
    }
    function progress_update()
    {
        progressBegin++;
        //如果开始标志已经大于结束标志
        if (progressBegin > progressEnd)
            progress_clear(); //清空进度条
        else
            //否则继续更新进度条
            document.getElementById("progress"+progressBegin).style.backgroundColor =
progressColor;
        //在一定的时间间隔内循环更新进度条
        progressTimer = setTimeout("progress_update()",progressInterval);
    }
    function linkto()
    {
        //显示 div
```



```
document.getElementById("framediv").style.visibility="visible";  
//调用更新进度条的方法  
progress_update();  
}  
</script>
```

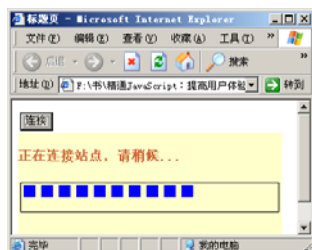


图 17-3 进度条的演示效果

即可。

【运行效果】

进度条的演示效果如图 17-3 所示。

【难点剖析】

进度条的设置需要两段代码实现，一段是进度条的更新，另一段是进度条的清空。进度条更新需要定时器“setInterval”，其可以设置每隔一段时间执行固定的 JavaScript 方法。清空进度条比较简单，只需要将进度条所在的 div 的“visibility”属性设置为“hidden”即可。

17.4 用 CSS+JS 制作进度条（二）

【实例描述】

本例学习如何使用 CSS+JavaScript 的方式实现常见的进度条。

【实现代码】

```
<script language="JavaScript" type="text/javascript">  
function pro_bwcheck(){  
    //检测浏览器的代码——通用  
    this.ver=navigator.appVersion  
    this.agent=navigator.userAgent  
    this.dom=document.getElementById?1:0  
    this.opera5=this.agent.indexOf("Opera 5")>-1  
    this.ie5=(this.ver.indexOf("MSIE 5")>-1 && this.dom && !this.opera5)?1:0;  
    this.ie6=(this.ver.indexOf("MSIE 6")>-1 && this.dom && !this.opera5)?1:0;  
    this.ie4=(document.all && !this.dom && !this.opera5)?1:0;  
    this.ie=this.ie4||this.ie5||this.ie6  
    this.mac=this.agent.indexOf("Mac")>-1  
    this.ns6=(this.dom && parseInt(this.ver) >= 5) ?1:0;  
    this.ns4=(document.layers && !this.dom)?1:0;  
    this.bw=(this.ie7 || this.ie6 || this.ie5 || this.ie4 || this.ns4 || this.ns6 ||  
this.opera5)  
    return this  
}  
var bw=new pro_bwcheck()  
numImages=12  
loaderWidth=350  
currentImg=0  
function pro_doc_size(){  
    //判断显示区域  
    this.x=0;this.x2=bw.ie && document.body.offsetWidth-20||innerWidth||0;
```

```

this.y=0;this.y2=bw.ie && document.body.offsetHeight-5||innerHeight||0;
if(!this.x2||!this.y2) return message('窗体的宽度和高度不够!');
this.x50=this.x2/2;this.y50=this.y2/2;
return this;
}
function pro_obj(obj,nest){ //定义页面中承载进度条的控件属性
    nest=(!nest)? "":'document.'+nest+'.'
    this.evnt=bw.dom? document.getElementById(obj):bw.ie4?document.all[obj]: bw.ns4?
eval(nest+"document.layers." +obj):0;
    this.css=bw.dom||bw.ie4?this.evnt.style:this.evnt;
    this.ref=this.css
    this.w=this.evnt.offsetWidth||this.css.clip.width||
        this.ref.width||this.css.pixelWidth||0;
    return this
}
pro_obj.prototype.moveIt = function(x,y){ //定义移动方法
    this.x=x;this.y=y; this.css.left=x;this.css.top=y
}
pro_obj.prototype.clipTo = function(t,r,b,l,setwidth){ //重新定义裁剪方法
    this.ct=t; this.cr=r; this.cb=b; this.cl=l
    if(bw.ns4){
        this.css.clip.top=t;this.css.clip.right=r
        this.css.clip.bottom=b;this.css.clip.left=l
    }else{
        if(t<0)t=0;if(r<0)r=0;if(b<0)b=0;if(b<0)b=0
        this.css.clip="rect("+t+", "+r+", "+b+", "+l+")";
        if(setwidth){this.css.pixelWidth=this.css.width=r;
            this.css.pixelHeight=this.css.height=b}
    }
}
var oLoad2
function startLoading(){
    page=new pro_doc_size()
    //以下设置加载控件的一些基本属性
    oLoadCont=new pro_obj('divLoadCont')
    oLoad=new pro_obj('divLoad1','divLoadCont')
    oLoad2=new pro_obj('divLoad2','divLoadCont.document.divLoad1')
    oLoadText=new pro_obj('divLoadText','divLoadCont.document.divLoad1')
    oLoad.moveIt(page.x50-loaderWidth/2,page.y50-20)
    oLoadText.moveIt(loaderWidth/2 - oLoadText.w/2,10)
    oLoad.clipTo(0,loaderWidth,40,0,1) //裁剪区域
    oLoad2.per = loaderWidth/numImages //百分比
}
function loadIt_display(ok){
    currentImg++
    if(oLoad2) oLoad2.clipTo(0,oLoad2.per*currentImg,40,0,1) //裁剪显示的区域
    if(currentImg<=numImages) setTimeout("loadIt_display(1)",200)
    //使用定时器不断加载图片

```



```
else{
    oLoadCont.css.visibility='hidden'           //进度条加载完成后隐藏
}
}
</script>
```

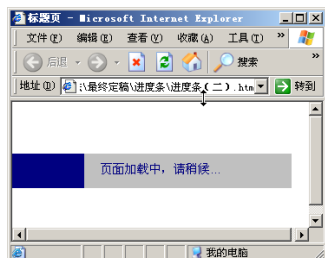


图 17-4 进度条的运行效果

【运行效果】

进度条的运行效果如图 17-4 所示。

【难点剖析】

本例的重点在于样式表中 div 层的布局。样式表设置 div 的背景色和边框颜色为蓝色，然后使用 JavaScript 计算进度条的宽度和每次显示的进度，其中重点是“clipTo”方法。本例重新定义了此方法，目的是实现一个矩形块状区域的显示。

17.5 进度条形式的下载效果

【实例描述】

在网页中加载数据或下载文件时，鉴于带宽问题可能需要经过一定的等待时间，此时可以提供一个加载或下载进度条提示用户等待时间。本例将学习如何制作一个下载进度条。

【实现代码】

```
SCRIPT language="javascript">
var NUMBER_OF_REPETITIONS = 40;
var nRepetitions = 0;
var g_oTimer = null;
//开始下载的方法
function startLongProcess()
{
    divProgressDialog.style.display = "";
    resizeModal();
    btnCancel.focus();
    // 设置窗口为大小可更改模式
    window.onresize = resizeModal;
    //当用户非正常中断时，添加一个提示
    window.onbeforeunload = showWarning;
    continueLongProcess();
}
function updateProgress(nNewPercent)
{
    // 更改进度条的进度
    divProgressInner.style.width = (parseInt(divProgressOuter.style.width)
        * nNewPercent / 100)+ "px";
}
//取消进度条的方法
```

```

function stopLongProcess()
{
    if (g_oTimer != null)
    {
        window.clearTimeout(g_oTimer);
        g_oTimer = null;
    }
    // Hide the fake modal DIV
    divModal.style.width = "0px";
    divModal.style.height = "0px";
    divProgressDialog.style.display = "none";

    // 移除窗体事件
    window.onresize = null;
    window.onbeforeunload = null;

    nRepetitions = 0;
}
//判断进度是否执行完毕的方法
function continueLongProcess()
{
    if (nRepetitions < NUMBER_OF_REPETITIONS)
    {
        var nTimeoutLength = Math.random() * 250;
        //进度条的进度
        updateProgress(100 * nRepetitions / NUMBER_OF_REPETITIONS);
        g_oTimer = window.setTimeout("continueLongProcess();", nTimeoutLength);
        nRepetitions++;
    }
    else
    {
        stopLongProcess();
    }
}
function showWarning()
{
    //用户非正常退出时的提示信息
    return "有一个应用程序正在运行，是否确定要退出";
}
function resizeModal()
{
    divModal.style.width = document.body.scrollWidth;
    divModal.style.height = document.body.scrollHeight;
    divProgressDialog.style.left = ((document.body.offsetWidth -
divProgressDialog.offsetWidth) / 2);
    divProgressDialog.style.top = ((document.body.offsetHeight -
divProgressDialog.offsetHeight) / 2);
}

```

</SCRIPT>

需要在 body 中用 div 层设计一个进度条样式的对话框，详细代码可参考随书光盘。

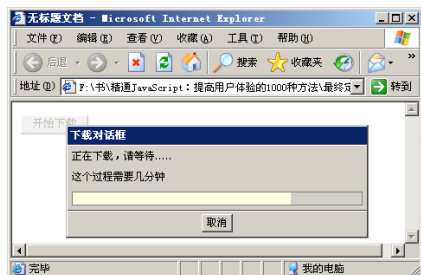


图 17-5 下载的进度提示界面

【运行效果】

下载的进度提示界面如图 17-5 所示。

【难点剖析】

本例的重点是如何判断进度条的进度，其中使用了语句“ $100 * nRepetitions / \text{NUMBER_OF_REPETITIONS}$ ”。“nRepetitions”变量相当于步长，在此处每增加一个进度“nRepetitions”变量会自增“1”。“NUMBER_OF_REPETITIONS”是一个常量，其值为“40”。

17.6 滑动条（一）

【实例描述】

滑动条用来显示一个百分比数据。本例通过一个类似刻度尺的组件实现滑动条的效果。

【实现代码】

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE>滑动条</TITLE>
    <META NAME="Generator" CONTENT="EditPlus">
    <META NAME="Author" CONTENT="">
    <META NAME="Keywords" CONTENT="">
    <META NAME="Description" CONTENT="">
    <script language="javascript">
      var objContainsDiv;
      var objTrackBar;
      var objTrackPath;
      var objScaleDiv;
      var scaleNumber = 20;
      var scaleLenth; //刻度长度
      var vLeft;
      var vaildLength; //滑块能够移动的有效长度
      function contentLoad()
      {
        objContainsDiv = trackDiv; //容器
        objTrackBar = createTrack();
        objTrackBar = objContainsDiv.appendChild(objTrackBar);

        objTrackPath = trackDegree;
        objTrackBar.onmousedown = trackBarBeforeMove;
        objTrackBar.onmouseup = trackBeforeMouseup;
```

```

objTrackPath.onclick = setPos;

vaildLength = parseInt(objContainsDiv.offsetWidth) -
parseInt (objTrackBar. offsetWidth) - 2;
scaleLenth = Math.round(parseInt(objContainsDiv.offsetWidth) /scaleNumber);

//手动生成刻度线
for(var i=0;i<scaleNumber - 1;i++){
    objScaleDiv = this.document.createElement("<div style ='position:
absolute;left:50;top:13;font-size:4pt;font-weight:lighter;color:#999999;width:3:'/>");
    objScaleDiv = objContainsDiv.appendChild(objScaleDiv);
    with(objScaleDiv){
        style.left = scaleLenth*(i + 1);
        innerText = "|";
    }
}

function createTrack(){
    //创建滑动条
    var objBarContainsDiv;
    objBarContainsDiv = this.document.createElement("<div style='position:
absolute;left:0;top:0;height:16;width:11;z-index:2;/>");

    //创建矩形区域
    var objBarTop = this.document.createElement("<div style='position:
absolute;left:0;top:0;height:10;width:11;font-size:1px;border-top:solid 1px #999999;
border-right:solid 1px #666666;border-left:solid 1px #cccccc;z-index:2;background:#cccccc;/>");
    objBarTop = objBarContainsDiv.appendChild(objBarTop);
    var objPointDiv;
    var iScale = 0;
    for(var i=0;i<6;i++){
        //创建红色标识区域，用来指向刻度
        objPointDiv = this.document.createElement("<div style='position:
absolute;background:red;font-size:1px;z-index:2;border-right:solid 1px #990000;/>");
        iScale = i + 1;
        with(objPointDiv){
            style.left = iScale;
            style.top = parseInt(objBarTop.style.pixelHeight) + (iScale - 1);
            if((parseInt(objBarTop.style.pixelWidth) - 2*iScale)<0){
                break;
            }
            style.width = parseInt(objBarTop.style.pixelWidth) - 2*iScale;
        }
        objPointDiv = objBarContainsDiv.appendChild(objPointDiv);
    }

    return objBarContainsDiv;
}

```



```

function setPos(){ //单击滑动条时，设置滑动条位置

    trackBeforeMove();
    trackLevel.innerText = Math.round(parseInt(objTrackBar.style.left)*
100/vaildLength) + "%"; //显示刻度数
}

function trackBarBeforeMove(){ //移动滑动条前的准备
    vLeft = window.event.x - objTrackBar.style.pixelLeft;
    objTrackBar.style.background = "#ddddd"; //背景
    objTrackBar.setCapture(); //鼠标
    objTrackBar.attachEvent("onmousemove", trackBeforeMove);
    //动态添加移动事件
}

function trackBeforeMove(){ //滑块移动中

    var leftPoint;
    var pointDividLength;
    var vMousePositionX;
    if((event.x - objContainsDiv.offsetLeft - 8) > vaildLength || event.
x<objContainsDiv.offsetLeft) return;

    vMousePositionX = parseInt(event.x) - objContainsDiv.offsetLeft;
    leftPoint = Math.floor(vMousePositionX/scaleLenth);
    //左边最近的点序号
    pointDividLength = leftPoint*scaleLenth + scaleLenth/2;
    window.status = "leftPoint:" + leftPoint + " [vMousePositionX:" +
vMousePositionX + " pointDividLength:" + pointDividLength + " ]";
    if(vMousePositionX < pointDividLength){
        //粘连到左边点
        objTrackBar.style.left = leftPoint*scaleLenth;
    }
    if(vMousePositionX > pointDividLength){
        //粘连到右边点
        objTrackBar.style.left = (leftPoint+1)*scaleLenth;
    }

    if(parseInt(objTrackBar.style.left)>vaildLength){
        //移到了右边界
        objTrackBar.style.left = vaildLength;
    }

    if(parseInt(objTrackBar.style.left)<0){
        //移到了左边界
        objTrackBar.style.left = 0;
    }
}

```



```

        trackLevel.innerText = Math.round(parseInt(objTrackBar.style.left)*
100/vaildLength) + "%";
    }

    function trackBeforeMouseup(){                //滑动结束，解除绑定
        if(parseInt(trackLevel.innerText.replace("%",""))>100){
            objTrackBar.style.left = vaildLength;
            trackLevel.innerText = "100%";          //不能超出最大刻度 100
        }else if(parseInt(trackLevel.innerText.replace("%",""))<0){
            objTrackBar.style.left = 0;
            trackLevel.innerText = "0%";            //不能小于最小刻度 0
        }
        objTrackBar.detachEvent("onmousemove", trackBeforeMove);
                                                //撤销事件绑定
        objTrackBar.style.background = "#cccccc"; //改变背景色
        objTrackBar.releaseCapture();             //释放鼠标
    }

</script>
</HEAD>
<BODY onload="contentLoad()">
    <div id="trackDiv" style="position:absolute;left:100;top:50;border: solid 0px
#cccccc;width:700;height:23;background:#ddddd;">
        <hr id="trackDegree" size="1" color="#cccccc" style="position:absolute;
top:16;height:3;border:groove 1px #eeeeee;background:#666666;z-index:1;">
    </div>
    <span id="trackLevel" style="position:absolute;left:100;top:30;width:50;
font-size:9pt;color:red;">0%</span>
</BODY>
</HTML>

```

【运行效果】

滑动条效果如图 17-6 所示。

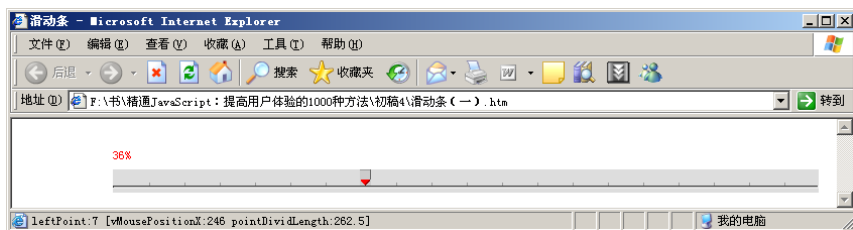


图 17-6 滑动条效果

【难点剖析】

本例的重点在于所有的刻度、滑动条、滑动条标识都是动态创建的。在“createTrack”方法中，“objBarTop”属性是创建的滑动条标识的上部分，“objPointDiv”是创建在滑动条下部的



红色指标，通过其可以轻松地看到滑动条的当前刻度。

17.7 滑动条（二）

【实例描述】

为了增加页面的美观性，可使用滑动条来显示百分比，而且还可以拖曳滑动条改变百分比的大小。本例学习如何实现这种功能的滑动条。

【实现代码】

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>滑块条</title>
<style type="text/css">
#trackBar
{
    background-color:#666666;
}

#trackBar_slider
{
    border:1px solid #808080;
    background-color:#FFFFFF;
}

#trackBar1
{
    background-color:blue;
}

#trackBar1_slider
{
    border:1px solid #808080;
    background-color:#FFFFFF;
}
</style>
</head>
<body>
<div id="info"></div>
<div id="trackBar1"> </div>
<script type="text/javascript" language="javascript">
//对象未创建完成之前，不能在函数之中用 this
function setTrackBar(trackBar, min, max, barPos)                //指定的div, 最小值, 最大值和位置
{
    this.trackBar = trackBar;                                    //承载滑动条的对象
    this.sliderIdStr = trackBar + "_slider";                    //滑动条
```

```

this.trackBarId = document.getElementById(this.trackBar);           //获取 div
this.sliderId = null;                                               //未创建滑动条对象
this.min = (min>=0)?min:0;                                          //最小值不能小于 0
this.max = (max>=min)?max:min;                                       //最大值必须大于最小值
this.barPos = (barPos>=min && barPos<=max)?barPos:min;             //位置必须在最大和最小之间

this.orientation = "h";                                           //设置对象和滑动条的位置
this.trackBarWidth = 100;
this.trackBarHeight = 10
this.sliderWidth = 10;
this.sliderHeight = 10;
this.Create = Create;
this.draging = false;
this.offset = 0;

this.BeforeDrag = BeforeDrag;                                     //绑定滑动事件
this.OnDrag = OnDrag;
this.EndDrag = EndDrag;
}
function Create(trackBar1)                                         //创建滑动条的方法
{
    this.trackBarId.innerHTML = "<div id=\"" + this.sliderIdStr + "\" + \" onmousedown="
    + "\"javascript:BeforeDrag(\" + trackBar1 + \");\" + \" style=\"position: relative;cursor:n-
    + "resize;\"></div>";
    this.sliderId = document.getElementById(this.sliderIdStr);
    this.sliderId.style.pixelTop = this.trackBarHeight - ((this.trackBarHeight-this.
    sliderHeight)*this.barPos)/(this.max-this.min) - this.sliderHeight;
    this.trackBarId.style.width = this.trackBarWidth;              //设置滑动条的初始位置
    this.trackBarId.style.height = this.trackBarHeight;
    this.sliderId.style.width = this.sliderWidth;
    this.sliderId.style.height = this.sliderHeight;

    return true;
}

var curTrackBar = null;

//准备拖曳
function BeforeDrag(trackBar)
{
    if (event.button != 1)                                         //如果不是鼠标左键，则返回
    {
        return;
    }
    document.body.style.cursor = "n-resize";                      //鼠标的样式
    curTrackBar = trackBar;
    curTrackBar.draging = true;
    curTrackBar.offset = curTrackBar.trackBarId.style.pixelTop + curTrackBar.sliderId.

```



```

style.pixelTop+curTrackBar.sliderId.offsetHeight- event.clientY;
    }

    function OnDrag() //实现拖曳的方法
    {
        if(!curTrackBar || !curTrackBar.draging)
        {
            return;
        }
        event.returnValue = false;
        var phyPos = 0;
        if (curTrackBar.orientation != "h")
        {
            phyPos = curTrackBar.trackBarId.style.pixelTop + curTrackBar.
trackBarId.offsetHeight - event.clientY - curTrackBar.offset;
            if (phyPos <= 0)
            {
                phyPos = 0; //如果拖动到最后端
            }
            else if(phyPos >= (curTrackBar.trackBarId.offsetHeight-curTrackBar. sliderId.
offsetHeight))
            {
                phyPos = curTrackBar.trackBarId.offsetHeight - curTrackBar. sliderId.
offsetHeight;
            }
            //改变滑动条的位置
            curTrackBar.sliderId.style.pixelTop = curTrackBar.trackBarId.offsetHeight -
phyPos - curTrackBar.sliderId.offsetHeight;
            curTrackBar.barPos =
parseInt(((curTrackBar.max-curTrackBar.min)*phyPos/(curTrackBar.trackBarId.offsetHeight-c
urTrackBar.sliderId.offsetHeight)));
        }

        OnTrackBarTxt();
    }
    function EndDrag() //结束拖曳
    {
        if (!curTrackBar)
        {
            return;
        }
        document.body.style.cursor = "default";
        curTrackBar.draging = false;
    }

    function OnTrackBarTxt() //拖曳时，改变滑动条的值
    {

```

```

        document.getElementById("info").innerHTML = curTrackBar.barPos+ " / " +
curTrackBar.max;
    }

    document.onmousemove = OnDrag;           //鼠标移动时，实现拖曳——开始滑动
    document.onmouseup = EndDrag;           //鼠标离开时，结束拖曳——即结束滑动
    var trackBarObj1 = new setTrackBar("trackBar1", 0, 100,100);
                                           //配置滑动条，指定最小值和最大值
    trackBarObj1.orientation = "v";         //滑动条的方法——垂直
    trackBarObj1.trackBarWidth = 15;        //对象的宽度
    trackBarObj1.trackBarHeight = 100;      //对象的高度
    trackBarObj1.sliderWidth = 15;         //滑动条的宽度
    trackBarObj1.sliderHeight = 10;        //滑动条的高度
    trackBarObj1.Create("trackBarObj1");    //创建滑动条
</script>
</body>
</html>

```

【运行效果】

滑动条的运行效果如图 17-7 所示。

【难点剖析】

本例提供了关于滑动条的 5 个常用方法。“setTrackBar”方法用来设置滚动条的最小值、最大值和位置等。“Create”方法用来创建滑动条的承载器和滑动条本身。“BeforeDrag”方法主要判断用户的操作是否为“拖曳”。“OnDrag”方法表示拖曳过程中滑动条的改变。“EndDrag”方法用来实现拖曳完成后的一些状态恢复。这些方法可以直接移植到其他项目中使用。



图 17-7 滑动条的运行效果

17.8 窗体滚动条随文字增加自动滚动

【实例描述】

当窗体中的内容超过窗体宽度时，默认会出现窗体的滚动条。但如果要在窗口中输入文本，且长度超过了窗体的宽度时，滚动条不一定随正在输入的文本自动滚动。本例的目的就是让滚动条跟随文本增加一起滚动。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<script>
var txtMsg = "往右";           //要显示的文本
                                //动态创建 span 标签
var text = document.createElement("<span style=position:absolute;z-index: 3;font:

```



```
12;left:10;top:10></span>");  
var stat =true;  
function writeText()  
{  
    text.innerHTML = txtMsg;           //显示文本  
    txtMsg += "继续往右";               //连接文本内容  
    div1.appendChild(text);            //将文本添加到 div 中  
    window.setTimeout("writeText()",400); //设置定时器，不间断显示文本  
    if (stat) {window.scrollTo(10000,0);} //主要控制滚动条的位置  
}  
document.onmousedown = function(){stat = false;}  
</script>  
<body onload="writeText()">  
<div id=div1 style=position:absolute;top:20;left:20;width:10000;cursor:default>  
</div>  
</body>  
</html>
```

【运行效果】

本例的运行效果如图 17-8 所示。

【难点剖析】

本例的重点是对滚动条位置的设置。“window.scrollTo”方法用来设置窗体中滚动条的位置，其自带两个参数，参数一表示 X 轴的偏移量；参数二表示 Y 轴的偏移量。

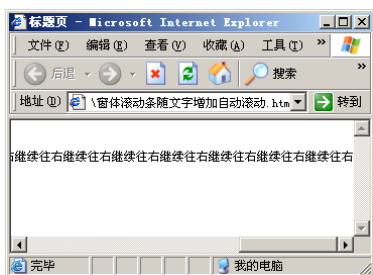


图 17-8 本例的运行效果

17.9 为 textarea 加横向滚动条

【实例描述】

textarea 用来输入多行文本，默认只有垂直滚动条。本例学习如何实现文本框的水平滚动条。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
</head>  
<body>  
<textarea wrap="off"></textarea>  
</body>  
</html>
```

【运行效果】

具有水平滚动条的文本框如图 17-9 所示。

【难点剖析】

本例的重点是 textarea 的“wrap”属性。此属性用来控制文本框是否换行。如果此值设为“off”，

则文本框内字符超过文本框的宽度时会出现水平滚动条,但不自动换行。如果此值设置为“soft”,则将出现 Word 中软回车的效果,即文本中字符串超过文本框宽度时文本会自动换行。



图 17-9 具有水平滚动条的文本框

17.10 记录滚动条位置

【实例描述】

当页面刷新时窗体中所有的内容会被重新初始化,但一些读书网站为了还原用户刷新前查看的内容,必须记录窗体滚动条的位置。本例介绍如何记录滚动条的位置。

【实现代码】

```
<script language="javascript">
    function SetCookie(sName, sValue)
    {
        date = new Date();
        s = date.getDate();
        date.setDate(s+1);
        document.cookie = sName + "=" + escape(sValue) + "; expires=" + date.toGMT-
String();
    }
    function GetCookie(sName)
    {
        var aCookie = document.cookie.split("; ");
        for (var i=0; i < aCookie.length; i++)
        {
            var aCrumb = aCookie[i].split("=");
            if (sName == aCrumb[0]) {
                return unescape(aCrumb[1]);
            }
        }
        return null;
    }
    function winLoad()
    {
        document.body.scrollLeft = GetCookie("scrollLeft");
        document.body.scrollTop = GetCookie("scrollTop");
    }
    function winUnload()
    {
        SetCookie("scrollLeft", document.body.scrollLeft)
        SetCookie("scrollTop", document.body.scrollTop)
    }
    window.onload = winLoad;
    window.onunload = winUnload;
</script>
```

**【难点剖析】**

本例的重点是如何保存滚动条的位置。页面中的滚动条分为水平滚动条和垂直滚动条。水平滚动条的位置使用“document.body.scrollLeft”获取，垂直滚动条的位置使用“document.body.scrollTop”获取。保存滚动条位置使用的是 Cookie，关于 Cookie 的操作可参考代码中的“SetCookie”和“GetCookie”方法。

17.11 彩色滚动条

【实例描述】

Windows XP 风格的界面可以呈现很多蓝色特效的页面，如标题、菜单、滚动条等。本例学习制作蓝色的滚动条。

【实现代码】

```
<style type="text/css">
body
{
    scrollbar-face-color: #b5daff;
    scrollbar-highlight-color: #ffffff;
    scrollbar-shadow-color: #000000;
    scrollbar-arrow-color: #0000ff;
    scrollbar-base-color: #6699ff;
    scrollbar-dark-shadow-color: #6699ff;
}
</style>
```

【难点剖析】

本例的重点就是窗体中文档的样式表。body 代表窗体中的文档部分，style 表示嵌入到当前窗体中的样式表，“scrollbar”用来设置滚动条的关键样式属性。

17.12 Windows XP 的滚动条

【实例描述】

使用 Windows XP 操作系统的用户都看到过 XP 的进度条，其不同于普通进度条，没有进度提示，只有来回渐隐渐现的滚动效果。本例学习如何制作这种效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<table border="0" cellpadding="0" cellspacing="0">
```



```

<tr>
  <td width="2" rowspan="2"></td>
  <td width="2" height="2"></td>
  <td height="2" bgcolor="#cccccc"></td>
  <td width="2"></td>
  <td width="2" rowspan="2"></td>
</tr>
<tr>
  <td width="2" height="2" bgcolor="#cccccc"></td>
  <td height="2"></td>
  <td width="2" bgcolor="#cccccc"></td>
</tr>
<tr>
  <td width="2" bgcolor="#cccccc"></td>
  <td width="2" height="2"></td>
  <td width="120 height="0"><marquee direction="right" scrollamount= "10"><table
style="font-size:1px;width:50px;height:10px;"
  <tr><td bgcolor="#e5fee5"></td><td bgcolor="#cbcdaf"></td><td bgcolor= "#8abf9a">
</td></tr></table></marquee></td>
  <td width="2" height="2"></td>
  <td width="2" bgcolor="#cccccc"></td>
</tr>
<tr>
  <td rowspan="2"></td>
  <td height="2" bgcolor="#cccccc"></td>
  <td height="2"></td>
  <td bgcolor="#cccccc"></td>
  <td rowspan="2"></td>
</tr>
<tr>
  <td height="2"></td>
  <td height="2" bgcolor="#cccccc"></td>
  <td></td>
</tr>
</table>
</body>
</html>

```

【运行效果】

本例的运行效果如图 17-10 所示。

【难点剖析】

本例的重点是在一个单元格中嵌套了 marquee 标签，实现了滚动特效。滚动的内容来自 marquee 标签中嵌套的表格“table”，里面实现了一个颜色的渐变效果。

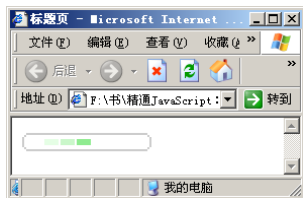


图 17-10 本例的运行效果

第 18 章 在线考题案例

本章导读

在线考试是目前比较流行的一种测试方法。这种考试不局限于某个地区，不需要复杂的监考，而且可以直接统计出考试结果。本章提供了几种常用的在线考题类型，包括单选、多选等。

18.1 在线考试代码（一）

【实例描述】

在线考试属于比较流行的测试方法，本例提供一个只有单选题的在线考试代码。

【实现代码】

```
<SCRIPT LANGUAGE="JavaScript">
function scoreCacu(form) {
    list=0
    if(form.Q1.value!=null && form.Q1.value=="*")
        {list=list+1}                                //如果返回的值带*, 则为正确, 分数值自增
    if(form.Q2.value!=null && form.Q2.value=="*")
        {list=list+1}
    if(form.Q3.value!=null && form.Q3.value=="*")
        {list=list+1}
    if(form.Q4.value!=null && form.Q4.value=="*")
        {list=list+1}
    if(form.Q5.value!=null && form.Q5.value=="*")
        {list=list+1}
    form.SCORE.value =eval(list)
        if (form.SCORE.value<=3){                    //5 题中, 分数值小于 3 为不及格
            ok="不及格! " }
    }

function clearScore(form) {                          //清空用户选择的项
    list=0
    form.Q1.value="?"
    form.Q2.value="?"
    form.Q3.value="?"
    form.Q4.value="?"
    form.Q5.value="?"
    form.SCORE.value =eval(list)
}

function msg() {                                     //发送邮件前的提示
    alert("感谢参加测试, 您的成绩已经将 mail 给 Webmaster。")
}
</SCRIPT>
```

【运行效果】

在线考试效果如图 18-1 所示。

【难点剖析】

本例的重点其实是如何判断正确的选择。本例在单选框的“onClick”事件中设置了选择项的值，如果选择正确则返回“*”。最后使用“scoreCacu”方法判断返回“*”的个数，如果小

于三个则属于不及格。

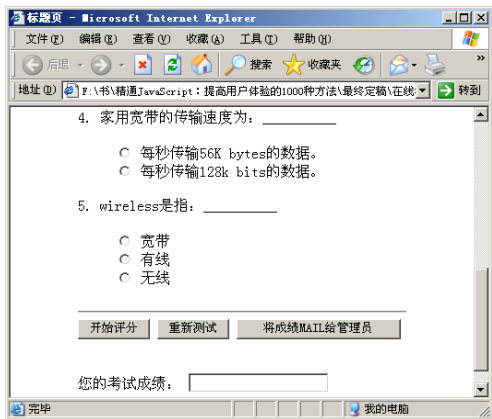


图 18-1 在线考试效果

18.2 在线考试代码（二）

【实例描述】

在线考试有很多种方法实现，本例通过一个包含正确答案的数组判断用户选择的答案与数组内容是否匹配，实现最终的考试效果。

【实现代码】

```
<script language="JavaScript">
var Total_Question = 4                                //考题的数量
var msg = ""

//正确答案——保存在数组中
var Answer = new Array(Total_Question)
Answer[0] = "TCP/IP"
Answer[1] = "网关"
Answer[2] = "Microsoft"
Answer[3] = "网络信息服务"

function GetSelectedButton(ButtonGroup)                //获取用户的选择
{
    for (var x=0; x < ButtonGroup.length; x++)
        if (ButtonGroup[x].checked) return x
    return 0
}

function ReportScore(correct)                            //显示最终的考试结果
{
    var SecWin =
        window.open("", "scorewin", "scrollbars,width=300,height=220")
    var MustHave1 = "<HTML><HEAD><TITLE>测验成绩报告</TITLE></HEAD><BODY>"
    var Percent = "<H2>测验成绩 : "+Math.round(correct/Total_Question*100)
        + "</H2><HR>"
}
```

```
lastscore=Math.round(correct/Total_Question*100)
if (lastscore == "100"){
    msg = MustHave1 +Percent + "<font color='red'>恭喜, 全部答对了! </font><p>" + msg + "<input
type='button' value='close' onclick=javascript:window.close()> </BODY></HTML>"
}
else {
    msg = MustHave1 +Percent + "<font color='red'>正确答案: </font><p>" + msg + "<input
type='button' value='close' onclick=javascript:window.close()></BODY> </HTML>"
}
}
SecWin.document.write(msg) //输出提示信息
msg = "" //清空信息提示
}

function Score()
{
    var correct = 0
    var wrong = 0
    for (number=0; number < Total_Question; number++)
    {
        var form = document.forms[number] //循环获取每一个问题
        var i = GetSelectedButton(form.q1) //获取问题中的选择项
        if (form.q1[i].value == "1") //如果选择 1, 则表示选择正确
        { correct++ }
        else
        { wrong++ //错误的变量+1
            msg += "Question "+(number+1)+". "
                +Answer[number]+"<BR>"
        }
    }
    ReportScore(correct) //输出最终结果
}
</script>
```

【运行效果】

在线考试效果如图 18-2 所示。答案提示效果如图 18-3 所示。

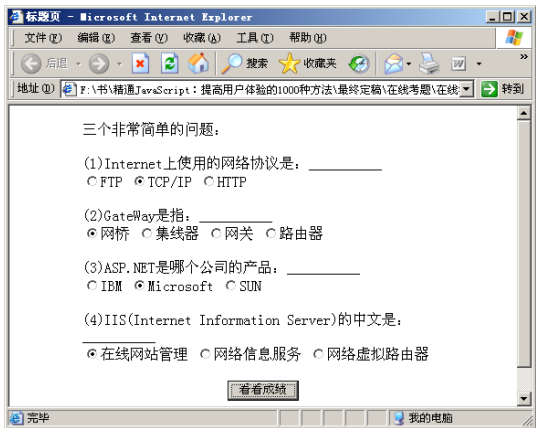


图 18-2 在线考试效果

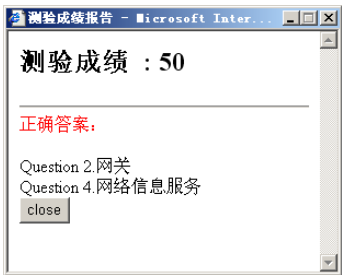


图 18-3 答案提示效果



【难点剖析】

本例的重点是将考题答案保存在一个数组“Answer”中，其目的是当用户选择错误时给出正确的答案。判断用户选择是否错误，是通过判断单选框的返回值，如果为“1”则表示选择正确。如果选择错误，通过“Answer[索引]”的方式获取正确的答案。

18.3 在线测试代码

【实例描述】

本例的代码实现一个类似于心理测试的网页。首先用户回答问题，然后判断用户的得分，根据得分实现一个心理测试的效果。

【实现代码】

```
<SCRIPT language=JavaScript>
function ResultEQ(form) {
var score = 0
if (form.Q01[0].checked) { score += -1 }           //判断返回值
if (form.Q01[1].checked) { score += 3 }
if (form.Q02[0].checked) { score += -1 }
if (form.Q02[1].checked) { score += 1 }
if (form.Q03[0].checked) { score += 3 }
if (form.Q03[1].checked) { score += 0 }
if (form.Q04[0].checked) { score += 3 }
if (form.Q04[1].checked) { score += 0 }
if (form.Q04[2].checked) { score += -1 }
if (form.Q05[0].checked) { score += -1 }
if (form.Q05[1].checked) { score += 3 }
if (form.Q06[0].checked) { score += 3 }
if (form.Q06[1].checked) { score += -1 }
if (form.Q07[0].checked) { score += 3 }
if (form.Q07[1].checked) { score += 0 }
if (form.Q08[0].checked) { score += 3 }
if (form.Q08[1].checked) { score += 0 }
if (form.Q09[0].checked) { score += 3 }
if (form.Q09[1].checked) { score += -1 }
if (form.Q10[0].checked) { score += 3 }
if (form.Q10[1].checked) { score += -1 }
score1=score*100/30                                //判断最终结果
if (score1 > 60) { alert('恭喜恭喜！你的成功率是'+score1+'好兆头，不是吗？')}
if (score1 > 30 && score1 <61) { alert('你的成功率是'+score1+'。怎么办？加把劲吧！')}
if (score1 < 31 ) { alert('你的成功率只有'+score1+'。你还是放弃吧...')}
}
</SCRIPT>
```

【运行效果】

在线测试的效果如图 18-4 所示。

【难点剖析】

本例很简单，通过“form.Q01[0].checked”判断用户选择的项，针对不同的选择项使“score”变量增加不同的值，最后根据用户的得分提示心理测试结果。

18.4 多选考试题

【实例描述】

随着网络的普及，在线考试的应用越来越广泛。本例学习如何制作包含多项选择的考试题。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<Script Language="javascript">
    function selectV(obj)
    {
        try{
            var oTd = obj.parentElement;           //获取表格的列
            var oTr = oTd.parentElement;           //获取表格的行
            var oTable = oTr.parentElement;         //获取表格
            var oTd_Answer = oTable.rows[oTr.rowIndex-1].cells[0].innerText;
                                                    //获取问题答案所在的单元格内容

            var oBegin = oTd_Answer.indexOf(" (") + 1; //获取“(”所在的位置
            var str = '';
            for(var i=0;i<oTd.children.length;i++){ //遍历问题选项
                if(oTd.children[i].tagName=='INPUT' && oTd.children[i].checked){
                    //如果选中
                    str += oTd.children[i].value;    //输出 value 值
                }
            }
                                                    //输出结束符号“(”
            oTable.rows[oTr.rowIndex-1].cells[0].innerText = oTd_Answer.
substring(0,oBegin) + str + ')';
        }catch(error){
            window.alert(error.description);        //显示错误信息
        }
    }
</Script>
</head>
<body>
<table>
    <tr>
        <td>1.你喜欢的城市()</td>
    </tr>
    <tr>
```

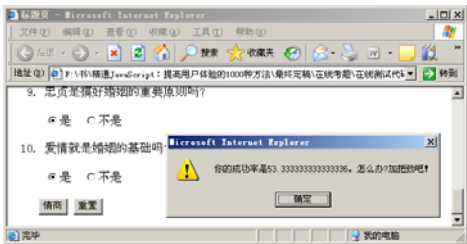


图 18-4 在线测试的效果



```
<td>
    <input type="checkbox" value="A" onclick="selectV(this);">A、北京<br>
    <input type="checkbox" value="B" onclick="selectV(this);">B、东京<br>
    <input type="checkbox" value="C" onclick="selectV(this);">C、纽约<br>
    <input type="checkbox" value="D" onclick="selectV(this);">D、首尔
</td>
</tr>
<tr>
    <td>2. 你喜欢的颜色 ( ) </td>
</tr>
<tr>
    <td>
        <input type="checkbox" value="A" onclick="selectV(this);">A、黑色<br>
        <input type="checkbox" value="B" onclick="selectV(this);">B、白色<br>
        <input type="checkbox" value="C" onclick="selectV(this);">C、红色<br>
        <input type="checkbox" value="D" onclick="selectV(this);">D、蓝色
    </td>
</tr>
</table></body>
</html>
```

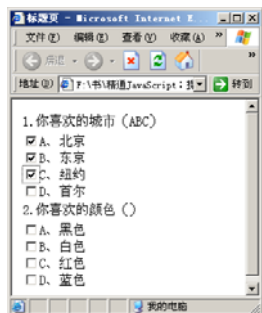


图 18-5 多项选择的效果

【运行效果】

多项选择的效果如图 18-5 所示。

【难点剖析】

本例熟练运用 DOM 实现单元格的获取。“parentElement”属性用来获取当前元素的父元素。获取单元格内容使用的是“cells[0].innerText”属性。本例还使用了字符串对象的一些操作方法。“indexOf”方法用来获取指定字符在字符串中的位置，“substring”方法用来截取从指定开始位置到指定结束位置之间的字符串。

18.5 在线心理测试脚本

【实例描述】

本例提供的代码可以为用户在线回答一个问题。用户选择一个项目，然后根据所选项给出选择结果的解释。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT language=JavaScript>
<!--
function processForm(form){
if (form.c1[0].checked==1) form.answer.value="喜欢纯洁的世界，自己内心一尘不染，可能有超凡脱俗
```



```
的性能。不太适合这个社会。";
    if (form.c1[1].checked==1) form.answer.value="踏实、稳重、有责任感。喜欢自己想问题，对身边的人也很关心";
    if (form.c1[2].checked==1) form.answer.value="稳重大方，喜欢大自然，心存善良，容易被骗！";
}
//-->
</SCRIPT>
</head>
<body>
<TABLE border=0 cellPadding=4 cellSpacing=0 height=78 width=500>
  <TBODY>
    <TR>
      <TD align=middle height=38 width=561><SPAN
        style="FONT-SIZE: 6pt">◆◆◆◆</SPAN><SPAN
        style="FONT-SIZE: 12pt"><STRONG>你是什么样的性格</STRONG></SPAN><SPAN
        style="FONT-SIZE: 6pt">◆◆◆◆</SPAN></TD></TR></TBODY></TABLE></DIV>

<FORM name=see>
<DIV align=left>
<TABLE border=0 cellPadding=10 cellSpacing=0 height=223 width=517>
  <TBODY>
    <TR>
      <TD height=1 width=517>
        <DIV align=left>
          <P>通过颜色判断，你最喜欢什么颜色</P></DIV>
          <DIV align=left>
            <P><INPUT name=c1 type=radio value=1>白色。<BR><INPUT name=c1 type=radio
              value=3>黑色。<BR><INPUT name=c1 type=radio value=V1>蓝色。
          </P></DIV></TD></TR>
    <TR>
      <TD align=middle height=27 width=517><INPUT onclick=processForm(this.form) style=
"FONT-FAMILY: 宋体; FONT-SIZE: 9pt" type=button value=已选好，看看你有什么样的性格!></TD></TR>
    <TR>
      <TD align=middle height=-12 width=517><TEXTAREA cols=47 name=answer rows= 5 style=
"COLOR: rgb(255,0,0); FONT-FAMILY: 宋体; FONT-SIZE: 9pt; LINE-HEIGHT: 15px"></TEXTAREA>
    </TD></TR></TBODY></TABLE></body>
</html>
```

【运行效果】

心理测试的运行效果如图 18-6 所示。

【难点剖析】

本例的重点是“processForm”方法。此方法获取用户选择的单选项，然后根据所选项返回不同的内容。“checked”属性用来判断单选控件是否被选择，值为“true”时表示此项被选中。

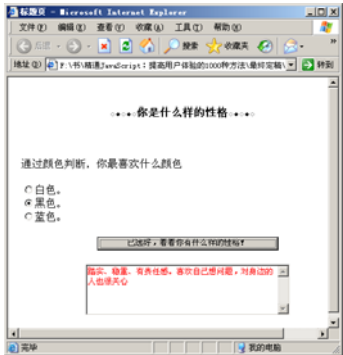


图 18-6 心理测试的运行效果



18.6 电脑检测健康情况

【实例描述】

网络上有很多测试身材的网页，通过用户输入的身高和体重计算用户的健康情况。本例以一个简单的实例学习如何判断身体的健康情况。

【实现代码】

```
<script LANGUAGE="JAVASCRIPT">
function ClearForm(form){                                //清空页面中的文本框
    form.weight.value = "";
    form.height.value = "";
    form.bmi.value = "";
    form.comment.value = "";
}
function bmi(weight, height) {                            //计算健康值
    bmindx=weight/eval(height*height);
    return bmindx;
}
function checkform(form) {
    if (form.weight.value==null||form.weight.value.length==0
        || form.height.value==null||form.height.value.length==0){//判断用户的输入
        alert("\n 请填写身高和体重! ");
        return false;
    }
    else if (parseFloat(form.height.value) <= 0||
        parseFloat(form.height.value) >=500||
        parseFloat(form.weight.value) <= 0||
        parseFloat(form.weight.value) >=500){
        alert("\n 输入的数据有问题 \n 请重新输入.");    //输入数据范围
        ClearForm(form);
        return false;
    }
    return true;
}
function computeHealth(form) {                            //根据健康值判断描述信息
    if (checkform(form)) {
        yourbmi=Math.round(bmi(form.weight.value, form.height.value/100));
        form.bmi.value=yourbmi;
        if (yourbmi >30) {
            form.comment.value="不要再闹了!!!";
        }
        else if (yourbmi >28 && yourbmi <=30) {
            form.comment.value="真的太胖了?";
        }
        else if (yourbmi >23 && yourbmi <=28) {
```

```
        form.comment.value="不是一般的胖啊?";
    }
    else if (yourbmi >21 && yourbmi <=23) {
        form.comment.value="您现在是偏胖,应该注意饮食!";
    }
    else if (yourbmi >=19 && yourbmi <=21) {
        form.comment.value="哇!!!您太苗条了! 是标准身材";
    }
    else if (yourbmi >=18 && yourbmi <19) {
        form.comment.value="您是不是营养不良?????";
    }
    else if (yourbmi >=17 && yourbmi <18) {
        form.comment.value="这个不是骨头吗?????";
    }
    else if (yourbmi <17) {
        form.comment.value="按照生物学来说这种生物是不能生存的!!!!!!";
    }
    }
    return;
}
</script>
```

【运行效果】

测试结果如图 18-7 所示。

【难点剖析】

本例的重点是对健康值的计算。当健康值为“20”时，表示一切正常。高于或低于“20”都算不太正常。“Math.round”方法是四舍五入的函数，用来获取一个整数。本例的算法其实不够精确，仅作参考。



图 18-7 测试结果

第 19 章

文件处理和打印的技巧

本章导读

JavaScript 虽然是客户端脚本语言，但可以与客户端各种文件交互，这需要通过专门的 ActiveX 组件。本章通过几个常用的案例，学习如何使用 JavaScript 与 Office 文档和文本文件交互。

19.1 判断上传文件的类型

【实例描述】

上传文件的时候，为了确保选择文件的正确性，需要判断文件的类型。本例使用正则表达式判断文件类型。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type=file onchange="alert(this.value.match(/^.*(\\.)\\.({1,8})$/) [3]+'类型')">
</body>
</html>
```

【运行效果】

类型的输出结果如图 19-1 所示。

【难点剖析】

本例的重点主要包括 file 上传控件的使用,选择文件后触发的事件,判断文件类型的正则表达式。file 上传控件可以弹出一个对话框,允许用户选择所有类型的文件。当用户选择完文件后触发“onChange”事件。正则表达式主要是通过对文件名中小数点的判断获取文件的类型。

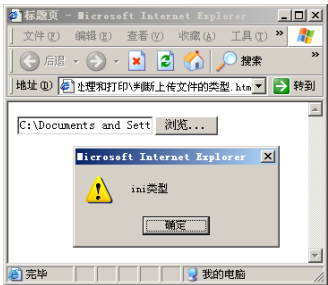


图 19-1 类型的输出结果

19.2 改变上传文件控件的样式

【实例描述】

file 控件是一个标准的 HTML 控件，可用来实现文件的选择和上传，但其样式过于单调，有时候用户会要求改变这种上传文件控件的样式。本例学习如何改变此控件的样式。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style type="text/css">
#myInput{border:1px solid #0000FF;}
#myBtn{width:90px;height:21px;font-size:12px;padding-top:3px;border-left:1px solid #FFFFFF;border-top:1px solid #FFFFFF;border-right:1px solid #666666; border-bottom:1px solid #666666}
</style>
```

```
</head>
<body>
<input type="text" id="myInput" > <input type="button" id="myBtn" value="选择上传的文件">
<input type="file" id="myfile" onchange="myInput.value=this.value" style="width:0;
position:absolute;left:43mm;filter:alpha(opacity=10)"></body>
</html>
```

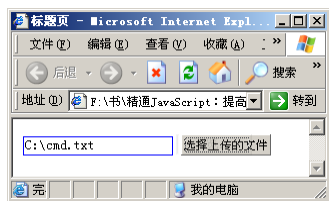


图 19-2 改变后的上传控件效果
内容即可。

【运行效果】

改变后的上传控件效果如图 19-2 所示。

【难点剖析】

本例利用了层的概念。将自己创建的按钮放在默认 file 按钮的上面，这样用户看到的就是带有样式的按钮，当用户单击此按钮时，实际上还是触发的 file 控件的事件，在此事件中设置文本框的内容即可。

19.3 上传文件一次完成

【实例描述】

默认的 HTML 控件 file 可以实现文件的上传，但必须经过选择文件和单击上传按钮两个步骤。本例学习如何实现一步上传文件的操作。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input id="myfile" type="file" style="display:none" onpropertychange="document.
getElementById('filename').innerText=this.value" />
<a href="javascript:document.getElementById('myfile').click()">浏览...</a>
<br /><span id="filename"></span>
</body>
</html>
```

【运行效果】

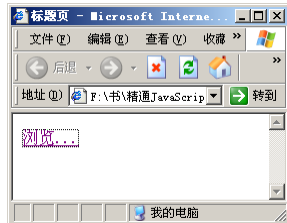


图 19-3 本例的运行结果

本例的运行结果如图 19-3 所示。

【难点剖析】

本例将 file 控件隐藏，然后使用 a 标签的“href”属性调用 file 控件的“click”方法实现文件的选择。当选择文件后，触发 file 控件的“onpropertychange”事件，在其中完成上传文件名的获取。

19.4 使用正则表达式判断文件扩展名

【实例描述】

在上传用户文件时，有时候需要限制文件类型。本例将通过一个简单的正则表达式，学习如何判断用户上传的文件类型是否合法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type="text" name="filetype" value="a.aspx">
<input type="button" value="检查" onclick="if(filetype.value.match(/\.(php| asp|aspx|
exe|com|bat|jpg|swf|gif|jsp|js|jsp|rar|zip)$/i)==null)alert('文件格式错误! ');else alert('格
式正确! ');">
</body>
</html>
```

【运行效果】

本例的运行效果如图 19-4 所示。

【难点剖析】

本例在正则字符串中对所有允许的文件类型进行了列举。“match”是正则的常用方法，用来判断字符串中是否包含指定的字符，如果没有则返回“null”。

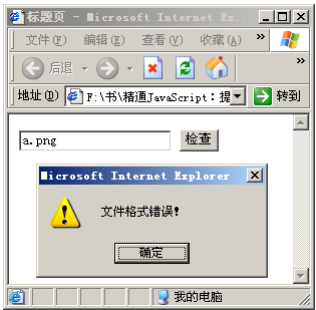


图 19-4 本例的运行效果

19.5 多附件上传效果

【实例描述】

在网络硬盘或邮箱附件功能中，常常需要上传大量的文件，为了提高上传速度，可允许用户同时选择多个文件，然后一次上传。本例就介绍实现此功能的方法。

【实现代码】

```
<script language="javascript" >
var n=0; //初始化数组索引为 0，之后随着变化
var fileCount=1; //总共输入了多少个有值的 file 控件，初始化为 1
var tempRow=0; //动态表格的临时行
var maxRows=0; //动态表格的临时列
var num = 1; //file 控件数组下标，从 1 开始，默认显示一个
var fileCount=1; //整个操作中，总共用了多少个 file 控件
function addFile()
{
var str = '<a href=#? class="addfile" id="a' + num + '"><input type= "file" size="50"
```



```

class="addfile" onchange="addFile()" name="uploadFile[' + num + '].file" ' + '>';//待插入
的文件控件

    var fileText;                //得到文件控件的值
    var ary;                      //分割文件，以'\ '号
    var fileTextValue;           //取出最后的文件名
    fileText = document.all("uploadFile[" + n + "].file").value;
    ary = fileText.split("\ ");
    fileTextValue = ary[ary.length-1];
    document.all("a" + n).style.display = "none";//将前一个 P 的子元素设为不可见

    //在前面一个 file 控件隐藏后，接着再在原来的位置上插入一个
    document.getElementById('MyFile').insertAdjacentHTML("beforeEnd",str);
    tempRow=fileTable.rows.length-1;    //fileTable 就是那个动态的 table 的 ID 了
    maxRows=tempRow;
    tempRow=tempRow+1;
    var Rows=fileTable.rows;    //Rows 数组
    var newRow=fileTable.insertRow(fileTable.rows.length);    //插入新的一行
    var Cells=newRow.cells;      //Cells 数组
    for (i=0;i<3;i++)            //每行两列数据，一列用来显示文件名，一列显示"删除"操作
    {
        var newCell=Rows(newRow.rowIndex).insertCell(Cells.length);
        newCell.align="center";
        switch (i)
        {
            case 0 : newCell.innerHTML="<td width='40%' align='left'><span id='"+n+"'>
</span></td>";
                break;
            case 1 : newCell.innerHTML="<td width='20%' align='left'><a href= 'javascript:
delTableFileRow(\"" +tempRow+ "\",\"" + n + "\" )'>删除</a></TD>";
                break;
            case 2 : newCell.innerHTML="<td width='40%' align='left'>&nbsp; </TD>";
                break;
        }
    }
    maxRows+=1;
    document.getElementById(n).insertAdjacentText("beforeBegin",fileTextValue);
    n++;
    num++;
    fileCount++;
}

function delTableFileRow(rowNum,fileCount)
{
    if (fileTable.rows.length >rowNum){
        fileTable.deleteRow(rowNum);    //删除当前行
    }else
    fileTable.deleteRow(fileTable.rows.length-1);
    //从元素 P 上删除子节点 a (与删除表格行同步)
    document.all("MyFile").removeChild(document.all("a" + fileCount)); fileCount--;
}

```



```
//总数 -1
}
</script>
```

【运行效果】

上传多个文件的效果如图 19-5 所示。

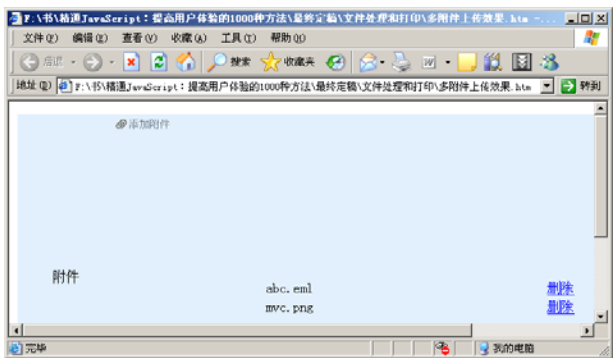


图 19-5 上传多个文件的效果

【难点剖析】

本例的难点是如何动态添加行以显示上传的文件，其中还使用了 file 控件来上传文件。动态添加行使用“insertRow”方法，添加列使用“insertCell”方法。

19.6 上传控件内容清空

【实例描述】

HTML 中提供的上传控件“file”，在选定上传文件后，文件名是不允许手动修改的，必须通过选择文件对话框实现修改。本例介绍如何使用 JavaScript 实现上传控件内容的手动清空。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function clearFile()
{
    var myfile = document.getElementById('file');
    myfile.outerHTML="<input type='file' id='file'>";
}
</script>
</head>
<body>
<input type="file" id="file" />
<input type="button" id="btn" value="清空" onclick="clearFile()" /></body>
</html>
```

【难点剖析】

本例的重点是“outerHTML”属性。“outerHTML”包括整个标签，而不仅限于标签内部的内容。如本例中的 file 控件的“outerHTML”就是整个 file 控件的代码“<input type="file" id="file" />”。

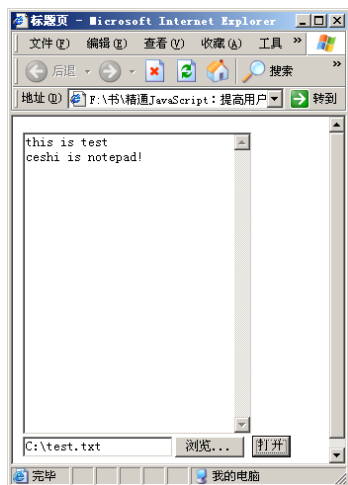
19.7 textarea 显示记事本文件的内容

【实例描述】

使用 JavaScript 的行为组件可以轻松实现界面与文本文件的交互。本例学习如何调用这些行为组件。

【实现代码】

```
<HTML XMLNS:IE>
<head>
<title>标题页</title>
</head>
<body>
<SCRIPT>
function onDownloadDone(s) {
    content.value=s;                                //设置文本框的内容
}
</SCRIPT>
<textarea name="content" cols=30 rows=20></textarea>
<div ID="oDownload" STYLE="behavior:url(#default#download)" />
<input type=file name='myfile'>
<input type=button value="打开" onclick="oDownload.startDownload(myfile. value,
onDownloadDone)">
</body>
</html>
```



【运行效果】

textarea 显示记事本文件的内容效果如图 19-6 所示。

【难点剖析】

本例的重点是“download”行为。行为可以在样式表中定义后使用“class”调用，也可以在标签中直接使用“style”属性来定义。定义行为时与其所在的标签几乎没有任何关系。“download”行为中有个“startDownload”方法，用来下载指定的文件。

图 19-6 textarea 显示记事本文件的内容效果

注意：行为是“behavior”的直译。

19.8 使用 FSO 读写文本文件

【实例描述】

文件操作是网页中常用的数据处理方法，有时可以将网页内容保存到文本文件或 XML 文件中。本例学习如何使用 JavaScript 操作文本文件。

【实现代码】

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>pubs</title>
<style>
table {
border:2 groove black;
position:absolute;
top:10;
left:10;
}
td {
border:1 ridge blue;
}
</style>
</head>
<script language="Javascript">
var path="c:\\"; //文件路径
var fname="test.txt" //文件名
window.status="邮件信息"; //状态栏信息
function getFileName(){
if (txtFile.value != "" && txtFile.value != " ") //如果用户不输入文件名
fname=txtFile.value; //使用默认文件名
}
function saveFile(){ //保存文件的方法
var fso,file;
if (txtContent.value == ""){ //判断是否填写了文件内容
alert("请输入文件内容!");
return;
}else{
getFileName(); //获取文件名
fso=new ActiveXObject("Scripting.FileSystemObject"); //创建文件对象
file = fso.CreateTextFile(path + fname,true); //指定文件详细路径
file.WriteLine(txtContent.value); //输出文件内容
file.close(); //关闭文件写入流
alert("保存完毕!");
```



```
    }  
  }  
  function readFile(){  
    //读取文件的方法  
    var fso,str,file;  
    getFileName();  
    fso = new ActiveXObject("Scripting.FileSystemObject"); //创建文件对象  
    str = "文件内容为空";  
    if (fso.FileExists(path + fname)){  
      //判断文件是否存在  
      file=fso.OpenTextFile(path + fname,1); //打开文件  
      str=file.readall(); //读取文件所有内容  
      file.close(); //关闭文件读取流  
    }  
    txtContent.value = str; //显示文件内容  
  }  
  
</script>  
<body>  
<table width="437" height="157" border="0" align="center" cellpadding="0" cellspacing="0">  
  <tr>  
    <td width="433" height="28">文件名:  
      <input type="text" id="txtFile">  
      <button name="save" onClick="Javascript:saveFile()">保存</button>  
      <button name="read" onClick="Javascript:readFile()">读取</button>  
    </td>  
  </tr>  
  <tr>  
    <td height="23"><div align="center">文件内容</div></td>  
  </tr>  
  <tr>  
    <td><textarea name="txtContent" rows="18" cols="60"></textarea></td>  
  </tr>  
</table>  
</body>  
</html>
```



图 19-7 本例的运行效果

【运行效果】

本例的运行效果如图 19-7 所示。

【难点剖析】

本例的难点是如何使用操作文件的“Scripting.FileSystemObject”组件。此组件就是常说的 FSO 对象，用于在 JavaScript 中处理文件。此对象的“CreateTextFile”方法用来创建文件，注意创建文件时需要指定文件的绝对路径。“OpenTextFile”方法用来打开文件，也需要文件的绝对路径。“WriteLine”方法用来写内容到文件。“readall”方法用来读取文件内容。

19.9 自动启动文件下载

【实例描述】

如果网站中提供下载功能，那么用户单击“下载”按钮后，如何自动弹出下载对话框，或者自动调用下载软件呢？本例将实现这种自动的文件下载功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type="button" name="down" value="download" onclick="javascript:window. location=
'http://localhost/myfile.rar';">
</body>
</html>
```

【运行效果】

本例启动调用下载软件的效果如图 19-8 所示。

【难点剖析】

如果本地机器上安装了下载软件（如蚂蚁、快车等），通常在下载文件时这些软件都会自动启用，这与软件的设置有一定关系。如果没有安装这些软件，则会自动调用 IE 默认的下载对话框。

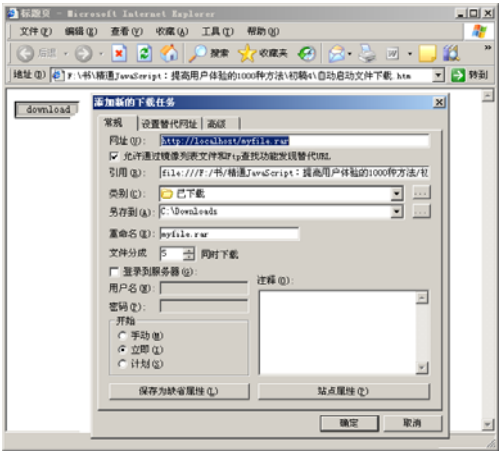


图 19-8 本例启动调用下载软件的效果

19.10 创建 Excel 文件

【实例描述】

JavaScript 可以在客户端创建文件，但这种方法一般不能轻易使用，因为客户端的浏览器安全设置可能不允许脚本随意创建文件。本例的代码仅供参考。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function createExcel()
{
    var ExcelSheet;
    ExcelApp = new ActiveXObject("Excel.Application"); //创建 Excel 对象
```

```
ExcelSheet = new ActiveXObject("Excel.Sheet");           //创建 Excel 中的一张工作表
ExcelSheet.Application.Visible = true;                   //Excel 可见
ExcelSheet.ActiveSheet.Cells(1,1).Value = "这是第一行第一列"; //设置第一行第一列的内容
ExcelSheet.SaveAs("C:\\mytest.xls");                      //保存 Excel 文件（有危险性）
ExcelSheet.Application.Quit();                            //退出 Excel 的应用程序
}
</script>
</head>
<body onload="createExcel()" >
</body>
</html>
```

【运行效果】
生成的 Excel 文件如图 19-9 所示。

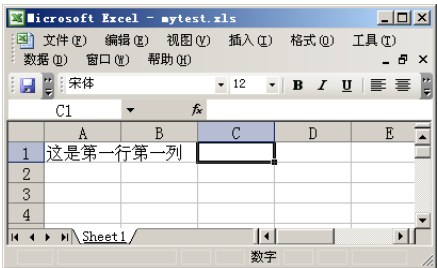


图 19-9 生成的 Excel 文件

【难点剖析】
本例的重点是如何创建 Excel 对象，如何操作 Excel 文档。整个实例的操作流程如图 19-10 所示。

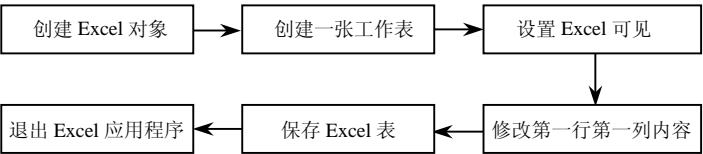


图 19-10 Excel 文件的创建流程

19.11 JavaScript 导出数据到 Excel

【实例描述】
IE 中提供了一些组件，用来在 JavaScript 中操作 Office 文档。本例学习如何将页面表格中的数据导入 Excel 中。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
```

```

<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function ExportExcel()
{
var oXL = new ActiveXObject("Excel.Application");    //创建 Excel 应用程序对象
var oWB = oXL.Workbooks.Add();                      //创建工作簿
var oSheet = oWB.ActiveSheet;                       //获取当前活动的工作簿
var table = document.all.data;                      //获取当前页面中的表格
var hang = table.rows.length;                       //获取表格有多少行
var lie = table.rows(0).cells.length;               //获取首行有多少列——多少标题

for (i=0;i<hang;i++)                                //添加标题到表格中
{
    for (j=0;j<lie;j++)
    {
        oSheet.Cells(i+1,j+1).Value = table.rows(i).cells(j).innerText;
                                                //设置标题的内容
    }
}
oXL.Visible = true;                                //设置 Excel 的属性
oXL.UserControl = true;
}
</SCRIPT>
</head>
<body>


```



【运行效果】

当前的页面数据如图 19-11 所示。导入 Excel 中的效果如图 19-12 所示。

【难点剖析】

本例的重点是如何创建 Excel 对象，以及如何将数据显示在 Excel 中。“Excel.Application”是 IE 中的“ActiveX”组件对象，可以创建 Excel 文件。“oSheet”变量是当前被激活的工作簿，使用“oSheet.Cells”可以获取工作簿中的单元格。



图 19-11 当前的页面数据

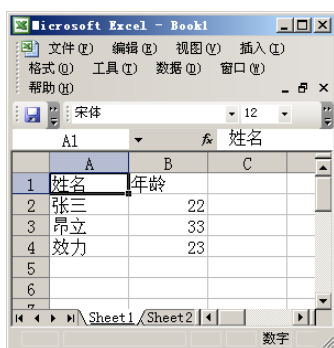


图 19-12 导入 Excel 中的效果

19.12 JavaScript 读取自身文件内的 XML

【实例描述】

JavaScript 可以从外部 XML 文件中轻松读取文件，同时 XML 文件也可以嵌套在当前网页中。本例学习如何读取嵌套在当前页内的 XML 文件。

【实现代码】

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="javascript">
window.onload=function()
{
    alert(xml1.XMLDocument.documentElement.xml) //使用 XML DOM 对象获取 XML 内容
}
</script>
</HEAD>
<xml id="xml1">
    <?xml version="1.0"?>
    <myxml>
        <root>
            <son1>aaaa</son1>
            <son2>bbbb</son2>
        </root>
    </myxml>
</xml>
```



```
<root>
  <son1>cccc</son1>
  <son2>dddd</son2>
</root>
</myxml>
</xml>
<BODY>
</BODY>
</HTML>
```

【运行效果】

输出的 XML 内容如图 19-13 所示。

【难点剖析】

本例的重点是“XMLDocument”对象。“xml”是一个专门用于加载 XML 文件或 XML 格式数据的标签。内部的 XML 标识是“<?xml version="1.0"?>”。“XMLDocument”对象用来获取 XML 文件中的所有元素对象。

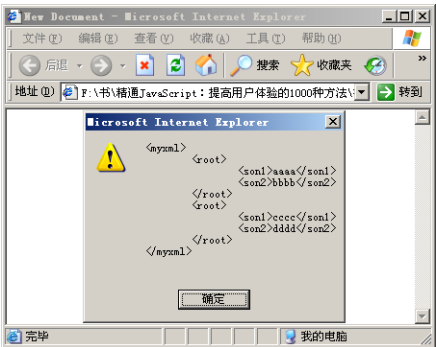


图 19-13 输出的 XML 内容

19.13 将 XML 文件绑定到 table

【实例描述】

JavaScript 可以动态加载 XML 文件，但 table 也提供一个属性可以将 XML 数据显示在表格中。本例学习如何将 XML 文件绑定到表格。

【实现代码】

```
<html>
<head>
<title>绑定到 Table</title>
</head>
<body>
<xml id="cdcat" src="将 XML 文件绑定到 Table.xml" tppabs="将 XML 文件绑定到 Table. xml"></xml>
<table border="1" datasrc="#cdcat">
<thead>
<tr><th>演唱者</th><th>歌名</th></tr>
</thead>
<tfoot>
<tr><th colspan="2">这是我喜欢的 CD</th></tr>
</tfoot>
<tbody>
<tr>
<td><span datafld="artist"></span></td>
<td><span datafld="title"></span></td>
</tr>
```

```
</tbody>
</table>
</body>
</html>
```

XML 文件的内容如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<cd>
  <cdcat>
    <artist>jay</artist>
    <title>她的睫毛</title>
  </cdcat>
  <cdcat>
    <artist>Eason</artist>
    <title>爱情呼叫转移</title>
  </cdcat>
</cd>
```



图 19-14 绑定表格后的效果

【运行效果】

绑定表格后的效果如图 19-14 所示。

【难点剖析】

本例的重点是 xml 标签，其用来加载指定的 XML 文件。table 标签的“datasrc”属性用来指定要绑定到 table 的数据源，与 xml 标签的“id”属性对应。在表格的单元格中，使用 span 标签的“datafld”属性对应 XML 文件中的字段。

19.14 使用 JavaScript 加载 XML 文件

【实例描述】

JavaScript 一般不能保存数据，使用 XML 可以方便地保存简单数据。本例学习如何使用 JavaScript 加载 XML 文件，并获取文件的内容。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
  <script type="text/javascript">
    var xmlhttp;
    function getData()
    {
      // 创建异步对象
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
      // 加载服务器——无参数
      xmlhttp.Open("GET","XMLFile.xml",true)
```

```

        //异步对象事件挂钩
        xmlhttp.onreadystatechange=stateChange;
        //发送请求——无参数
        xmlhttp.Send(null);
    }
    function stateChange()
    {
        if(xmlhttp.readyState==4 && xmlhttp.status==200)
        {
            //获取所有返回的数据
            var data=xmlhttp.responseText;
            //显示结果
            document.getElementById("divlist").innerHTML=data;
        }
    }
</script>
</head>
<body>
    <table style=" text-align:center">
        <tr>
            <td style="text-align:center">
                显示从 XML 文件中获取的数据</td>
        </tr>
        <tr>
            <td style="text-align:center">
                <input id="Button1" type="button" value="获取表格" onclick= "getData()"
/></td>
        </tr>
        <tr>
            <td style="text-align:center">
                <div id="divlist">
                </div>
            </td>
        </tr>
    </table>

</body>
</html>

```

本例使用的 XML 文件的代码如下所示：

```

<?xml version="1.0" encoding="utf-8" ?>
<table border="1">
    <tbody>
        <tr>
            <th>Name</th>
            <th>Sex</th>
            <th>Age</th>
        </tr>

```



```
<tr>
    <th>lisi</th>
    <th>male</th>
    <th>23</th>
</tr>
<tr>
    <th>wangwu</th>
    <th>male</th>
    <th>27</th>
</tr>
</tbody>
</table>
```

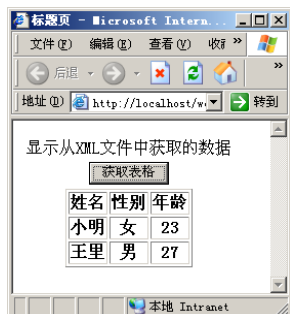


图 19-15 XML 文件显示的效果

【运行效果】

XML 文件显示的效果如图 19-15 所示。

【难点剖析】

本例的重点是 msxml2.domDocument 控件。首先使用“new ActiveXObject”创建此组件，然后使用其中的“load”方法。加载指定的 XML 文件，最后使用 DOM 获取文件中的内容。

19.15 动态加载 JavaScript 文件

【实例描述】

在当前页面中可以调用其他的 JavaScript 文件。本例就学习如何实现对外部文件的调用。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
    document.write('<script src=\'动态加载 JavaScript 文件 js\'></script>');//动态加载另//
外一个文件
</script>
</head>
<body>
</body>
</html>
```

【运行效果】

本例的运行效果如图 19-16 所示。

【难点剖析】

如果想在本网中嵌套一个 JavaScript 文件，需要使用代码“<script src=abc.js />”。这就可以在当前页面中动态输出这段代码，实现文件的调用功能。

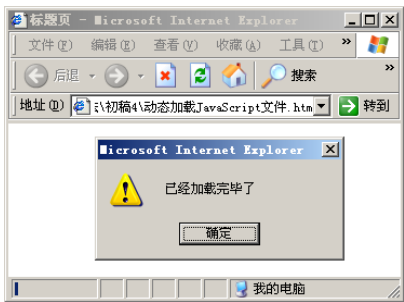


图 19-16 本例的运行效果

19.16 防止 JavaScript 文件被其他站直接引用

【实例描述】

由于 JavaScript 文件的安全很难保障，所以经常会出现随意调用别人网站 JavaScript 文件的情况。本例学习如何防止其他网站引用自己网站的 JavaScript 文件。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language=javascript>
function sts()
{
    if(window.location.hostname!="www.baidu.com")
    {
        alert("请不要剽窃本站的 js 文件!");
        window.top.location.href = "http://www.baidu.com/";
    }
}
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例使用“window.location.hostname”判断当前打开的页面是否属于自己的网站，如果不是则给出提示，并将页面导航到自己的网站。

19.17 检查机器是否安装 Word

【实例描述】



有时候需要用户下载一些帮助文档，为了保证文档顺利打开，可先判断用户的计算机中是否安装了 Word 程序。本例学习如何使用 JavaScript 判断机器中 Word 的安装情况。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
    function CheckWord()
    {
        try
        {new ActiveXObject("WScript.Shell");}           //创建新控件
        catch(x) {return false;}
        try
        {new ActiveXObject("Word.Application");}         //创建 Word 新应用程序
        catch(x){return null;}
        return true;
    }
    res=CheckWord();                                   //检查是否安装了 Word
    switch(res)
    {
        case true:
            alert("安装了 Word"); break;
        case null:
            alert("没有安装 Word"); break;
        case false:
            alert("当前机器的 ActiveX 被禁用");
        }
    }
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例中先使用“new ActiveXObject("WScript.Shell")”判断机器是否支持 ActiveX；然后使用“new ActiveXObject("Word.Application")”判断是否安装了 Word 程序。如果要判断 Excel 程序，可使用“Excel.Application”对象。

19.18 打印当前页

【实例描述】

网页默认情况下不支持水晶报表等专业打印工具。如果要打印网页则只能调用 JavaScript 中的打印方法，但可实现的打印效果非常有限。本例介绍最简单的打印方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
if (window.print) {
document.write('<form>' + '<input type=button name=print value="打印页面" '
+ 'onClick="javascript:window.print()"></form>');
}
</script>
</head>
<body>
</body>
</html>
```

【运行效果】

调用打印后的页面效果如图 19-17 所示。

【难点剖析】

本例的重点是 JavaScript 中的打印方法“window.print”。这是 IE 自带的打印功能，可自动调用 Windows 系统中默认的打印机，缺点是打印的数据量大时无法实现分页。

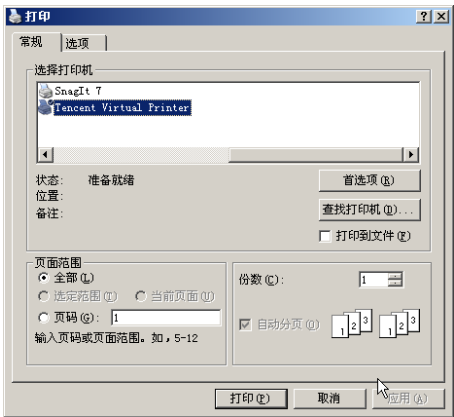


图 19-17 调用打印后的页面效果

【实例描述】

为了让用户了解要打印的内容，可在打印文档之前设置一个打印预览功能。本例介绍如何实现文档的打印预览。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<OBJECT classid="CLSID:8856F961-340A-11D0-A96B-00C04FD705A2" height=0 id=wb name=wb
width=0></OBJECT>
<input type=button value=打印预览 onclick="wb.execwb(7,1)">
<table style="width: 300px">
<tr>
<td>
阿</td>
<td>
不</td>
<td>
才</td>

```

```
</tr>
<tr>
  <td>
    的</td>
  <td>
    恶</td>
  <td>
    发</td>
</tr>
<tr>
  <td>
    的</td>
  <td>
    发</td>
  <td>
    的</td>
</tr>
</table>
</body>
</html>
```

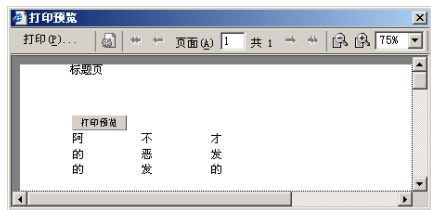


图 19-18 预览效果

件在本文档中的唯一标识 id。

【运行效果】

预览效果如图 19-18 所示。

【难点剖析】

本例的重点是 Object 组件。Object 组件是 IE 的一种内置控件，可通过“<OBJECT>”元素实现对组件的调用。调用组件时需要注明组件的“classid”属性，还要设置组件在本文档中的唯一标识 id。

19.20 隐藏不想打印的页面内容

【实例描述】

Web 页面的打印通常通过调用 IE 自带的打印功能，但此功能的缺点是只能打印页面中所有内容。如何只打印网页的一部分，而去掉那些不需要打印的内容呢？本例用一个变通的方法实现了局部页面的打印。

【实现代码】

```
<HTML>
<HEAD>
<TITLE>去除不想打印出的页面元素</TITLE>
<script type="text/javascript">
function preview()
{
  // 获取页面内容
```



```

var bdhtml=document.body.innerHTML;
var beginstr="<!--startprint-->";
var endstr="<!--endprint-->";
//获取要打印的内容
var prnhtml=bdhtml.substr(bdhtml.indexOf(beginstr)+17);
prnhtml=prnhtml.substr(0,prnhtml.indexOf(endstr));
//预览
window.document.body.innerHTML=prnhtml
//打印
window.print()
}
</script>
</HEAD>
<BODY background="" leftMargin=0 topMargin=0 rightMargin=0 bottomMargin=0 style=
"BACKGROUND-POSITION: center 50%">
<!--startprint-->
<DIV align=center>
<span>这里是我需要的内容</span>.....
</DIV>
<!--endprint-->
<div align="center">
<span>这里不是我我要的内容</span><br />
<input type="button" value="打印" onclick="preview()" />
</div>
</BODY>
</HTML>

```

【运行效果】

本例的初始运行效果如图 19-19 所示。单击“打印”按钮后的效果如图 19-20 所示。

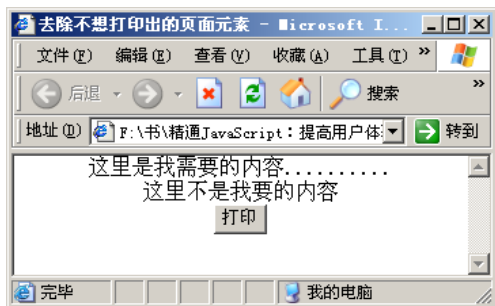


图 19-19 初始运行效果

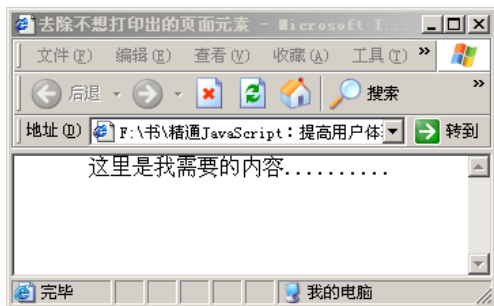


图 19-20 单击“打印”按钮后的效果

【难点剖析】

本例主要应用的是字符串内容的截取。在页面中设置两个标记，标记在所需内容的开始和结束位置。首先使用“document.body.innerHTML”获取页面中所有内容，然后利用字符串的“substr”方法截取需要的内容，并更改页面的 body 内容。最后调用“window.print”方法打印修订后的页面。



19.21 使用 ExecWB 直接打印

【实例描述】

实现打印的方法有两种：window.print 和 ExecWB 组件。本例将学习 ExecWB 的使用方法。

【实现代码】

```
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<title>无标题文档</title>
</head>
<body>
<OBJECT classid=CLSID:8856F961-340A-11D0-A96B-00C04FD705A2 height=0 id= WebBrowser
width=0></OBJECT>
<input name=Button onClick=document.all.WebBrowser.ExecWB(6,6) type=button value=直接打印>
</body>
</html>
```

【难点剖析】

本例中首先使用 OBJECT 标签加载 ExecWB 组件，注意加载时必须指定组件在本页中的唯一标识 id。“classid”是组件在组件库中的唯一标识。“ExecWB(6,6)”表示对当前页面进行打印。ExecWB 是通过参数中的数值来指定打印效果。

19.22 动态绑定 XML 文件

【实例描述】

绑定 XML 文件的内容可以在 span 中添加“datafld”属性，但这是静态的方法。本例学习如何使用 JavaScript 实现 XML 文件的动态绑定。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<XML ID="xmlUser">
  <?xml version="1.0" ?>
  <userinfo>
    <datacol userName="张三" userAge="20"/>
    <datacol userName="李四" userAge="21"/>
  </userinfo>
</XML>
<div id=mydiv>
```

```
</div>
<script language="JavaScript">
function xml2div(){
    var oXML;
    oXML = xmlUser.selectNodes("userinfo/dataacol");           //找到 XML 文件下的节点
    for(var i=0; i<oXML.length;i++){
        var oDiv = document.createElement("DIV");             //动态创建 div
        oDiv.setAttribute("userName",oXML[i].getAttribute("userName")); //为 div 绑定属性
        oDiv.setAttribute("userAge",oXML[i].getAttribute("userAge"));   //绑定属性
        oDiv.innerText = "Div: " + i;                               //设置 div 里面的内容
        mydiv.appendChild(oDiv);                                     //将 div 添加到指定的 div 下
    }
    alert(mydiv.innerHTML);
}
xml2div();
</script>
</body>
</html>
```

【运行效果】

本例运行的效果如图 19-21 所示。

【难点剖析】

本例中使用 XML 标签内嵌了一个 XML 文件, 然后使用“selectNodes”调用这个文件中的节点。使用“for”语句遍历所有的节点, 然后将节点添加到 div 中。

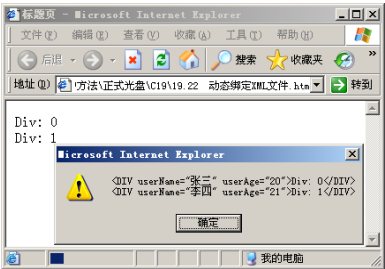


图 19-21 本例运行的效果

19.23 Kill Excel 的进程

【实例描述】

使用 JavaScript 调用 Office 的应用程序后会有些残留进程驻留在操作系统中。本例以 Excel 为例, 介绍如何 Kill 这些残留的进程。

【实现代码】

```
<HTML>
<head>
<title>无标题</title>
</head>
<BODY>
<INPUT type="button" value="关闭 Excel" name=btn1 onclick="StartExcel()">
<SCRIPT LANGUAGE=Javascript>
    var idTmr = "";
    function StartExcel() {
        var oExcel;
        oExcel = new ActiveXObject("Excel.Application");           //创建 Excel 对象
```



```
oExcel.Quit(); //关闭 Excel
oExcel = null; //释放对象
idTmr = window.setInterval("QuitExcel();",1); //定时清除
}
function QuitExcel() {
    window.clearInterval(idTmr); //清除定时器
    CollectGarbage(); //kill 进程
}
</SCRIPT>
</BODY>
</HTML>
```

【难点剖析】

本例的重点是“CollectGarbage”方法，通常被称为 GC。此方法用于清理当前 IE 中的“失效的对象示例”，也就是调用对象的析构方法。

第 20 章

调用操作系统的应用

本章导读

JavaScript 可以调用客户端操作系统的一些程序，听起来这是一件很危险的事情，因为大部分的防火墙和 IE 安全设置都禁止这种操作。本章只是提供几个常用的小例子，基本上都是一些相对安全的调用。



20.1 JavaScript 操作剪贴板

【实例描述】

JavaScript 提供的剪贴板有两种，系统自带的剪贴板和 `dataTransfer` 对象内部的剪贴板。本例将演示使用 Windows 系统自带的剪贴板。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <script language="javascript">
    function windowCopy(){
      window.clipboardData.setData("Text",txt1.innerHTML);
    }
    function windowParse(){
      var txt= window.clipboardData.getData("Text");
      txt1.innerHTML=txt+"粘贴完毕"
    }
  </script>
</head>
<body>
  <textarea cols=30 rows=10 id="txt1">
</textarea>
  <table id="mytbl" width="300" height="50" border="0" cellpadding="2" cellspacing="0"
bgcolor = "#FFb609">
    <tr>
      <td> <input id="Button1" type="button" value="window 剪贴板-剪切" onclick= "windowCopy()"
/></td>
      <td> <input id="Button1" type="button" value="window 剪贴板-粘贴" onclick= "windowParse()"
/></td>
    </tr>
  </table>
</body>
</html>
```

【运行效果】

操作剪贴板的运行效果如图 20-1 所示。

【难点剖析】

本例的重点是剪贴板的使用方法，剪切和粘贴。“`setData`”用来保存数据到剪贴板，“`getData`”用来获取剪贴板中的数据，使用时注意这两个方法的参数。

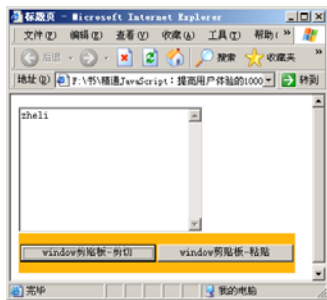


图 20-1 操作剪贴板的运行效果

20.2 打开硬盘驱动器

【实例描述】

本例在网站中并不多见，一般用于 FTP 上传文件，或者在服务器中打开文件。实现的效果是在 IE 中直接打开 Windows 操作系统的资源管理器。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<form action="file:///c|/"><input type="submit" value="打开 C 盘"></form>
<form action="file:///d|/"><input type="submit" value="打开 D 盘"></form>
<form action="file:///e|/"><input type="submit" value="打开 E 盘"></form>
</body>
</html>
```

【运行效果】

打开 C 盘的效果如图 20-2 所示。

【难点剖析】

本例的重点是如何从 IE 直接转到硬盘驱动器,这通过在浏览器中输入“file:///c|/”完成。默认 IE 的传输协议为“http://”。

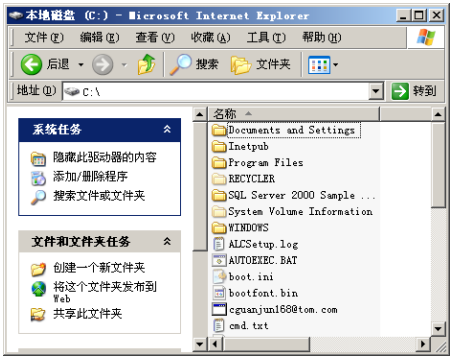


图 20-2 打开 C 盘的效果

20.3 单击加入收藏夹

【实例描述】

为了方便用户操作，可以提供用户收藏本网站的功能。本例以搜索网站为例，学习如何让用户收藏本站。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<span style="CURSOR: hand" onClick="window.external.addFavorite('http: //www.google.com', '
最爱的搜索')" title="Google 搜索">收藏搜索网站</span>
</body>
</html>
```



【运行效果】

单击“收藏搜索网站”后的效果如图 20-3 所示。

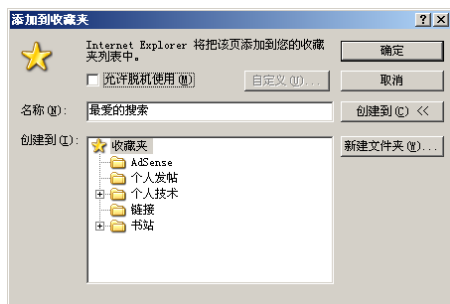


图 20-3 单击“收藏搜索网站”后的效果

【难点剖析】

本例的重点是“window.external.addFavorite”方法。其中“external”是关于系统文件操作的命令。“addFavorite”方法将指定的 URL 地址添加到收藏夹中，其第二个参数表示显示在收藏夹中的网站标题。

20.4 复制标题和网址

【实例描述】

复制标题和网址有利于用户方便地拷贝本页面，并推荐给好友，一般用于求职招聘网站。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
</head>
<body>
<input type="button" name="Submit" onClick='toClipBoard()' value="复制网址，传给好友">
<script language="javascript">
function toClipBoard()
{
var clipBoardContent="";
clipBoardContent+=document.title;           //获取页面标题
clipBoardContent+="\n";
clipBoardContent+=this.location.href;       //获取页面地址
//将拷贝内容存放到剪贴板中
window.clipboardData.setData("Text",clipBoardContent);
alert("复制成功，可以粘贴到你的QQ/MSN上，推荐给你的好友！");
}
</script>
</body>
</html>
```


【运行效果】

运行效果如图 20-4 所示。

【难点剖析】

本例的重点是如何将页面信息保存到剪贴板中。其中剪贴板“clipboardData”是 window 对象的一个属性。保存内容到剪切板的语法如下所示：

```
clipboardData.setData("唯一标识",存放的内容);
```

20.5 关闭输入法

【实例描述】

很多输入内容只允许是数字或英文，如身份证号或邮箱地址等。为了防止用户非法输入，除了使用正则表达式外还可以关闭操作系统的输入法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
</head>
<body>
<input id="txtname" style="ime-mode:disabled">
</body>
</html>
```

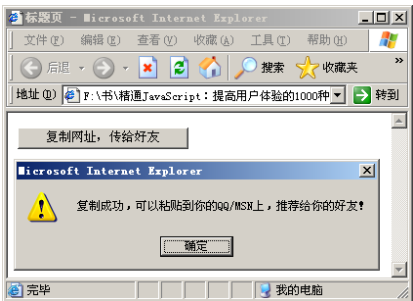


图 20-4 运行效果

因为本例的运行效果无法使用图像显示，所以未给出运行效果。

【难点剖析】

本例对 input 元素使用了样式，重点是“ime-mode”样式，其代表操作系统的输入法，设置其值为“disabled”表示禁止使用。默认状态下输入法是打开的。

20.6 检测屏幕分辨率

【实例描述】

早期版本的网页需要根据屏幕分辨率来设计界面，为了保证界面的完整效果，需要判断用户屏幕的分辨率，并根据分辨率调整页面。本例介绍如何判断用户屏幕的分辨率。

【实现代码】

```
<script language="JavaScript">
function getScreen()
{
  if ((screen.width == 640) && (screen.height == 480))
```



```
size = "640 x 480";
else if ((screen.width == 800) && (screen.height == 600))
size = "800 x 600";
else if ((screen.width == 1024) && (screen.height == 768))
size = "1024 x 768";
else if ((screen.width == 1280) && (screen.height == 1024))
size = "1280x1024";
else
size = "默认值为 640 x 480";
alert("屏幕分辨率为 " + size );
}
</script>
```

需要在 body 中添加一个按钮，用来调用“getScreen”方法，代码如下所示：

```
<input id="Button1" type="button" value="获取屏幕分辨率" onclick="getScreen()" />
```

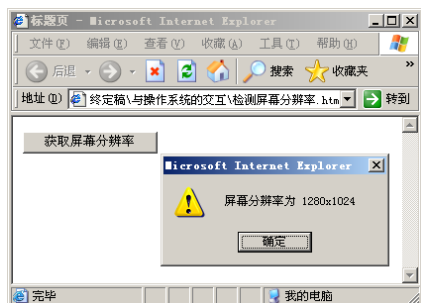


图 20-5 提示分辨率的界面

【运行效果】

提示分辨率的界面如图 20-5 所示。

【难点剖析】

本例通过 screen 对象来获取屏幕的高度和宽度，从而达到获取分辨率的目的。screen 对象表示用户的屏幕，用来获取与屏幕有关的所有信息。

20.7 检测系统信息

【实例描述】

不同的浏览器所支持的内部对象库会有不同（如 XMLHTTP 对象），为了调用正确的对象需要提前判断浏览器或系统的类型。本例显示浏览器版本、使用语言等。

【实现代码】

```
<script language="JavaScript">
    document.write("浏览器名称: "+navigator.appName+"<br>");
    document.write("浏览器版本号: "+navigator.appVersion+"<br>");
    document.write("系统语言: "+navigator.systemLanguage+"<br>");
    document.write("系统平台: "+navigator.platform+"<br>");
    document.write("浏览器是否支持 cookie: "+navigator.cookieEnabled+"<br>");
</script>
```

【运行效果】

检测系统信息的运行效果如图 20-6 所示。

【难点剖析】

本例的重点是 navigator 对象，其主要提供系统和浏览器的相关属性。“appname”属性表示浏览器的名称，可用其判断用户的浏览器是否支持特殊对象。



图 20-6 检测系统信息的运行效果

20.8 显示本地计算机信息

【实例描述】

获取本地计算机的信息是一件有安全隐患的操作，在运行这些代码前，IE 一般会弹出提示，询问是否允许获取本地计算机信息。本例演示如何获取本地计算机的信息。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
    var WshNetwork = new ActiveXObject("WScript.Network"); //创建
    alert("用户域: " + WshNetwork.UserDomain); //显示用户所在的域
    alert("计算机名称: " + WshNetwork.ComputerName); //显示计算机名
    alert("用户名: " + WshNetwork.UserName); //显示登录用户名
</script>
</head>
<body >
</body>
</html>
```

【难点剖析】

本例的重点是“WScript.Network” ActiveX 组件。此组件是“script host”提供的 COM 对象。其主要提供网络连接和远程打印机管理的函数。

20.9 检测浏览器浏览过的站点数

【实例描述】

浏览器对浏览过的网站都有一个缓存功能，可通过浏览器中的“历史记录”了解以前访问的站点。但本例是检测当前浏览器自打开后访问过的站点总数。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script language="javascript">
    var times=history.length;
    document.write(' 此浏览器已经去过'+times+'个站了。');
</script>
</body>
</html>
```

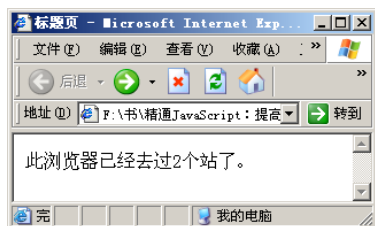


图 20-7 检测结果

【运行效果】

检测结果如图 20-7 所示。

【难点剖析】

本例的重点是 JavaScript 中的“history”对象。此对象可以操作浏览器返回上一次访问的站点，功能类似于浏览器工具栏中的“前进”、“后退”按钮。“history”对象提供一个“length”属性，可以获取当前打开的浏览器访问的站点总数。

20.10 IE 文件菜单中的“打开”命令

【实例描述】

有时候在应用程序中需要调用文件菜单中的命令，以方便用户操作。本例学习如何使用 JavaScript 调用菜单命令。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<a href="#" onclick=document.execCommand("open")>打开</a>
</body>
</html>
```

【运行效果】

运行“打开”命令后的效果如图 20-8 所示。

【难点剖析】

本例的重点是网页中执行命令的“execCommand”方法。其类似于 ADO.NET 中的数据库

操作命令，用来执行操作系统的一个动作，如本例中打开一个窗口。

20.11 打开“Internet 选项”对话框

【实例描述】

有时用户的浏览器设置不允许访问或下载本站点的内容，此时需要用户打开 Internet 选项进行调整。为了方便用户操作，本例学习如何在网页中让用户直接打开 Internet 选项对话框。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<a href="#" onclick=window.external.showBrowserUI("PrivacySettings",null) >internet 选项</a>
</body>
</html>
```

【运行效果】

打开“Internet 选项”对话框的效果如图 20-9 所示。

【难点剖析】

本例的重点是 external 对象，其用来操作文件系统的一些命令。“showBrowserUI”方法表示打开浏览器的一些工作界面，“PrivacySettings”表示打开的是选项对话框。

20.12 打开 Windows 系统的画板

【实例描述】

随着网络的盛行，网络安全一直是一个很大的问题。因为 JavaScript 脚本可以访问本地计算机上的文件，并能调用机器上的一些可执行程序。本例虽然只学习如何调用本地机器上的程序，但希望读者能去考虑一个问题，如何去禁止这种客户端的操作。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT language=JavaScript>
function RunPaint(){
```

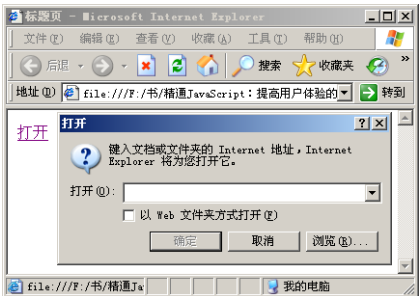


图 20-8 运行“打开”命令后的效果



图 20-9 打开“Internet 选项”对话框的效果



```
try{
    var objShell = new ActiveXObject("wscript.shell");    //创建 wscript.shell 对象
    objShell.Run('mspaint');                             //打开操作系统的画板
    objShell = null;                                     //释放资源
}
catch(e){
    alert("找不到画板文件");                             //捕获错误
}
}
</SCRIPT>
</head>
<body>
<input type=button value="打开画板" onclick="RunPaint()">
</body>
</html>
```

【难点剖析】

本例的重点是 wscript.shell 组件。wscript.shell 是一个 ActiveX 组件，其方法“Run”用来创建新的进程，该进程用指定的窗口样式执行指定的命令，本例的命令是打开系统的画板。

20.13 弹出保存对话框

【实例描述】

如果要保存网页，必须调用 IE 自带的“另存为”对话框，但有时候为了让用户下载一些文档，可调用操作系统的“保存”对话框。本例学习如何在网页中弹出保存对话框。

【实现代码】

```
<html>
<iframe src="" width="0" height="0" id="myIframe" name="myIframe"></iframe>
<textarea id="content" rows="6" cols="50"></textarea>
<P>
<button onclick="javascript:saveFile()">保存</button>
<script language="javascript">
function saveFile(){
var obj = window.frames("myIframe");                //找到窗体上的 iframe
    obj.document.open();                             //打开 iframe
    obj.document.write (document.getElementById("content").value);    //获取文档的内容
    obj.document.close();                             //关闭 iframe
    obj.document.execCommand("SaveAs");               //调用保存对话框
}
</script>
</html>
```

【运行效果】

本例的初始运行效果如图 20-10 所示。在文本框中输入内容后，单击“保存”按钮，打开

的保存对话框如图 20-11 所示。

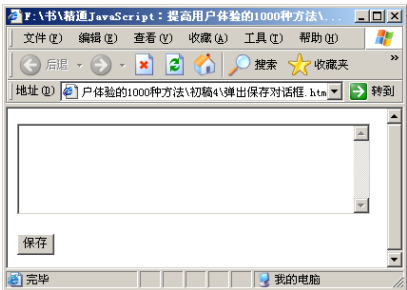


图 20-10 本例的初始运行效果

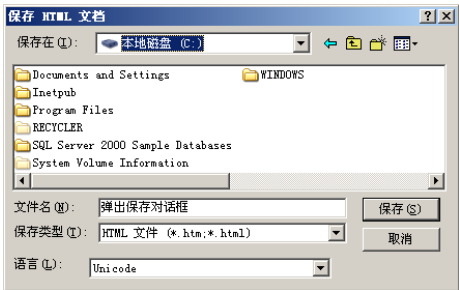


图 20-11 打开的保存对话框

【难点剖析】

本例的重点是“execCommand”方法，其一般用于浏览器与常用操作的交互，如撤销、重复、全选、保存等。其所带的参数不能随意更改，每一个参数对应一种操作。

20.14 进入页面弹出收藏夹

【实例描述】

很多网站为了推广各种服务，一般会建议用户将网页添加进收藏夹。本例通过一个简单的实例，学习如何自动弹出收藏夹。

【实现代码】

```
<SCRIPT LANGUAGE="JavaScript">
var expDays = 7;
url = "http://www.google.com/";
title = "收藏夹特效";
// 获取 Cookie 中保存的信息
function GetCookie (name)
{
    var arg = name + "=";
    var alen = arg.length;
    var clen = document.cookie.length;
    var i = 0;
    while (i < clen) {
        var j = i + alen;
        if (document.cookie.substring(i, j) == arg)
            return getCookieVal (j);
        i = document.cookie.indexOf(" ", i) + 1;
        if (i == 0) break;    }
    return null;
}
// 保存信息到 Cookie 中
```



```
function SetCookie (name, value)
{
var argv = SetCookie.arguments;
var argc = SetCookie.arguments.length;
var expires = (argc > 2) ? argv[2] : null;
var path = (argc > 3) ? argv[3] : null;
var domain = (argc > 4) ? argv[4] : null;
var secure = (argc > 5) ? argv[5] : false;
//设置 Cookie 的属性, 有效期、域、路径等
document.cookie = name + "=" + escape (value) +
((expires == null) ? "" : ("; expires=" + expires.toGMTString())) +
((path == null) ? "" : ("; path=" + path)) +
((domain == null) ? "" : ("; domain=" + domain)) +
((secure == true) ? "; secure" : "");
}
//删除指定的 Cookie
function DeleteCookie (name)
{
var exp = new Date();
exp.setTime (exp.getTime() - 1);
var cval = GetCookie (name);
document.cookie = name + "=" + cval + "; expires=" + exp.toGMTString();
}
var exp = new Date();
exp.setTime(exp.getTime() + (expDays*24*60*60*1000));
function amt(){
var count = GetCookie('count')
if(count == null) {
SetCookie('count','1')
return 1
}
else {
var newcount = parseInt(count) + 1;
DeleteCookie('count')
SetCookie('count',newcount,exp)
return count
}
}
function getCookieVal(offset)
{
var endstr = document.cookie.indexOf (";", offset);
if (endstr == -1)
endstr = document.cookie.length;
return unescape(document.cookie.substring(offset, endstr)); //解密 Cookie 中的信息
}
```



```
function checkCount()
{
var count = GetCookie('count');
//如果已经在收藏夹中添加了网站，则下一次不再出现添加收藏夹界面
if (count == null) {
count=1;
SetCookie('count', count, exp);
    if ((navigator.appName == "Microsoft Internet Explorer") && (parseInt (navigator.
appVersion) >= 4))    {
        window.external.AddFavorite (url,title);                //打开收藏夹的关键语句
    }
}
else {
    count++;
    SetCookie('count', count, exp);
}
}
checkCount();
</script>
```

【运行效果】

弹出收藏夹的效果如图 20-12 所示。

【难点剖析】

本例的重点是如何弹出收藏夹，并保证收藏夹中默认的名称和地址都是本站点。JavaScript 中的 window 对象提供了“external.AddFavorite”方法，用来打开收藏夹。AddFavorite 方法包含两个参数，收藏夹的标题和该收藏夹导航到的地址。

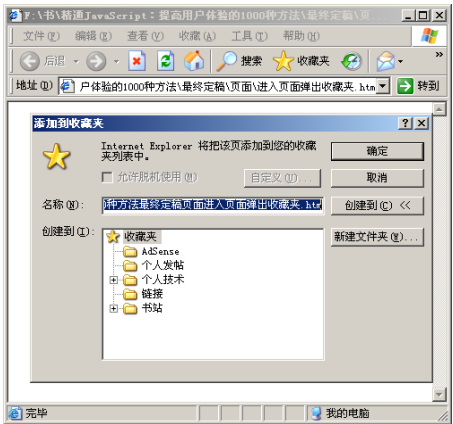


图 20-12 弹出收藏夹的效果

20.15 执行客户端的可执行程序

【实例描述】

本例的代码存在安全隐患，默认情况下浏览器是不允许直接运行可执行文件的。本例只是学习如何使用 JavaScript 执行可执行文件。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function exec (command) {
    window.oldOnError = window.onerror;                //错误处理的默认事件
    window._command = command;                        //要执行的命令
}
```



```
window.onerror = function (err) { //重新设置错误处理事件
    if (err.indexOf('Automation') != -1) { //如果因权限问题被终止
        alert('命令已经被用户禁止! ');
        return true;
    }
    else
        return false;
}
var wsh = new ActiveXObject('WScript.Shell'); //创建可执行应用程序对象
if (wsh)
    wsh.Run(command); //执行指定的命令
window.onerror = window.oldOnError; //恢复默认的错误处理事件
}
</script>
</head>
<body>
<a href="#" onclick="exec('C:/NTDETECT.COM')">执行文件</a>
</body>
</html>
```

【难点剖析】

本例的重点是“WScript.Shell”ActiveX 组件。其实现对可执行文件的调用，这些可执行文件包括操作系统中的记事本、写字板等，也包含一些“.BAT”、“.COM”文件。

20.16 自动调用 Outlook 发送邮件

【实例描述】

这是一个很简单的发送邮件实例，主要功能是指定邮件的标题和内容后，代码自动生成一封 Outlook 新邮件。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
</head>
<body>
<a href="mailto:cguanjun@tom.com?subject=测试邮件&body=这是一封测试的邮件">
发送邮件</a>
</body>
</html>
```

【运行效果】

生成的邮件界面如图 20-13 所示。

【难点剖析】

本例的重点是代码“mailto:cguanjun@tom.com?subject=测试邮件&body=这是一封测试的邮

件”。其中“mailto”表示邮件发送到的目的地；“subject”表示邮件的标题；“body”表示邮件的内容。

20.17 弹出窗口选择颜色

【实例描述】

IE 中有个用来选择颜色的窗口组件。本例学习如何调用这个组件, 以及如何使用这个组件实现颜色的选择。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function getColor(color)
{
    var sInitColor = color;                                // 获取参数传递的颜色
    if (sInitColor == null || sInitColor=="")
        var sColor = myColor.ChooseColorDlg();             // 打开颜色对话框
    else
        var sColor = myColor.ChooseColorDlg(sInitColor);    // 设置颜色
        sColor = sColor.toString(16);                       // 转换为 16 进制颜色
    if (sColor.length < 6) {                                 // 如果颜色小于 6 位
        var sTempString = "000000".substring(0,6-sColor.length); // 格式化为 6 位
        sColor = sTempString.concat(sColor);
    }
    sColor = "#" + sColor;                                   // 添加颜色标签
    return sColor;
}
</script>
</head>
<body>
<input type="text" name="txt1" value="这里显示最终调用的颜色">
<input type="button" value="选取颜色" onClick="txt1.value=getColor()">
<OBJECT id=myColor CLASSID="clsid:3050f819-98b5-11cf-bb82-00aa00bdce0b" width="0px"
height="0px"></OBJECT>
</body>
</html>
```

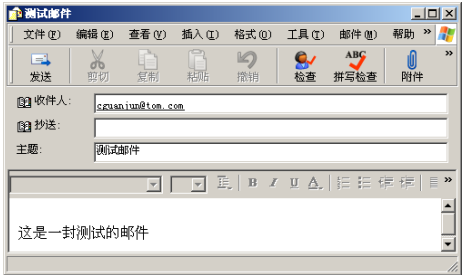


图 20-13 生成的邮件界面

【运行效果】

调用颜色选择窗口的效果如图 20-14 所示。选择颜色后的效果如图 20-15 所示。

【难点剖析】

本例的重点是 myColor 组件的调用。方法“ChooseColorDlg”用来打开颜色选择器。选择

的颜色是字符型的，需要使用“toString(16)”转换为 16 进制，然后使用“format”方法格式化为最终的颜色编码。



图 20-14 调用颜色选择窗口的效果

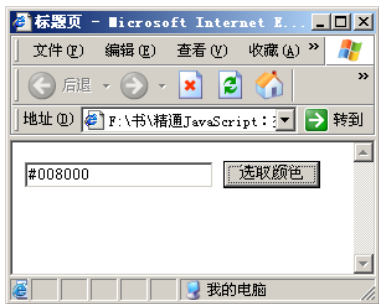


图 20-15 选择颜色后的效果

20.18 弹出框式邮件发送

【实例描述】

本例使用 JavaScript 的弹出式窗口“prompt”实现邮件地址和邮件主题的输入，然后通过 Outlook 发送邮件。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script>
var author="张三";
if (author == "张三"){
    phrompt=prompt;
    snarkconf=confirm;
}
function mailsome1(){
    who=phrompt("输入朋友的邮箱: ", "zhangsan@263.net"); // 邮件地址
    what=phrompt("输入邮件主题: ", "[no subject]"); // 邮件主题
    if (snarkconf("确定要发信给 "+who+"; 主题是 "+what+"")==true) // 确定是否要发信
    {
        parent.location.href='mailto:'+who+'?subject='+what+''; // 调用 Outlook
    }
}
```

```
}  
</script>  
<a href='javascript:mailsome1()'>发信给朋友</a><form>  
<input type=button value=发信!\" onclick=\"mailsome1()\"></form>  
</body>  
</html>
```

【运行效果】

弹出式的输入对话框如图 20-16 所示。

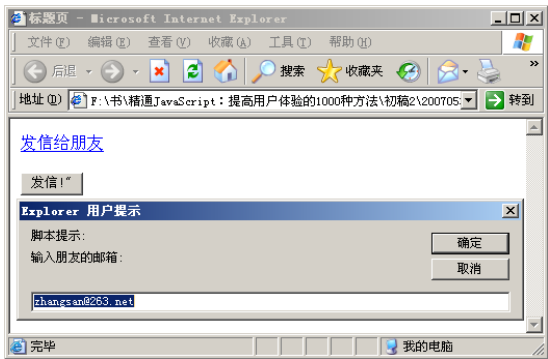


图 20-16 弹出式的输入对话框

【难点剖析】

本例的重点是“prompt”方法，其可以弹出一个对话框，用户可在其中的文本框内输入数据，并捕获用户的输入。根据这些输入的值设置邮件的地址和主题，然后使用 Outlook 发送邮件。

20.19 把网站作为用户的 Active 桌面

【实例描述】

如果喜欢某个网站的页面，非常想把它设置为自己的桌面，那么可以学习一下本例的方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
<script language="javascript">  
function setDesktop()  
{  
    //将百度设置为桌面  
    window.external.AddDesktopComponent("http://www.baidu.com","website",  
0,0,800,600); }  
</script>  
</head>
```



```
<body>
<INPUT onclick="setDesktop()" type="button" value="点这里看看是什么效果" name="Button2" />
</body>
</html>
```

【运行效果】

将百度设置为桌面后的效果如图 20-17 所示。



图 20-17 将百度设置为桌面后的效果

【难点剖析】

本例中主要是利用“external”的“AddDesktopComponent”方法，此方法的使用语法如下所示。

```
external.AddDesktopComponent(url,type,0,0,800,600);
```

其中各个参数的意义分别是 URL、type、起始位置 X 坐标、起始位置 Y 坐标、高度和宽度。其中类型“type”有两种：“image”和“website”。

20.20 判断是否安装了 Flash 插件

【实例描述】

为了保证服务器的 Flash 文件可以正常运行，需要先判断客户端是否安装了 Flash 插件。如果没有安装则进行提示。

【实现代码】

```
<HTML>
<head>
<title>无标题</title>
```

```
</head>
<BODY>
<body>
<SCRIPT LANGUAGE="JavaScript">
    //创建 AcitiveXObject 组件
var swf = new ActiveXObject('ShockwaveFlash.ShockwaveFlash');
    //判断组件的值
(swf) ? document.writeln('你已经安装了插件') : document.writeln('你没有安装插件');
</SCRIPT>
</body>
</HTML>
```

【难点剖析】

本例的重点是 ShockwaveFlash.ShockwaveFlash 组件。首先使用“new”关键字创建一个新的组件对象，然后使用三元运算符“?:”判断对象是否创建成功，如果创建不成功，则表示未安装 Flash 插件。

第 21 章

流行技术：DOM 和 userData 的应用技巧

本章导读

JavaScript 虽然没有其刚刚兴起时那么炙手可热，但现在的 Web 流行技术依然离不开客户端脚本的支持。随着技术研究的深入，一些能提高开发性能的技术被挖掘出来，本章主要介绍两种技术，DOM 和 userData。其中 DOM 是文本对象模型，以树形结构的方式访问文档。userData 是一种客户端保存数据的方式，类似于 Cookie。

21.1 使用 userData 保存文本内容

【实例描述】

userData 行为提供了一个比 Cookie 更具有动态性和容量更大的数据结构。本例学习用其充当 Cookie 的方法。

【实现代码】

```
<HTML>
<HEAD>
<STYLE>
    .userData {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
function saveInput(){
    var oPersist=Form1.myText;
    oPersist.setAttribute("sPersist",oPersist.value);
    //将 oPersist.value 存储为 sPersist 属性
    oPersist.save("oXMLBranch");    //存储在名为 oXMLBranch 的 userData 存储区
}
function loadInput(){
    var oPersist=Form1.myText;
    oPersist.load("oXMLBranch");    //载入在名为 oXMLBranch 的 userData 存储区
    oPersist.value=oPersist.getAttribute("sPersist");
    //将 sPersist 属性赋值给 oPersist.value
}
</SCRIPT>
</HEAD>
<BODY>
<FORM ID="Form1">
<INPUT CLASS="userData" TYPE="text" ID="myText">
<INPUT TYPE="button" VALUE="加载" onclick="loadInput()">
<INPUT TYPE="button" VALUE="保存" onclick="saveInput()">
</FORM>
</BODY>
</HTML>
```

【运行效果】

先在文本框内输入姓名，然后单击“保存”按钮，再清空文本框，单击“加载”按钮后的效果如图 21-1 所示。

【难点剖析】

userData 使用“setAttribute”方法设置变量的值，使用“save”方法保存到指定名称的存储区域。如果要从 userData 中找到需要的变量，则先使用“load”方法找到指



图 21-1 单击“加载”按钮后的效果



定名称的存储区域，然后使用“getAttribute”找到对应的变量值。

21.2 使用 userData 保存 select 标记

【实例描述】

如果用户选择下拉框内容后关闭了浏览器，为了能在重新打开页面时正确显示用户的选择，通常先使用 Cookie 保存用户的选择。本例学习如何用 userData 替换 Cookie 实现这种客户端保存功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style>
    .userData {behavior:url(#default#userData);}
</style>
</head>
<body>
    随便选择一项，重新启动页面，看是否保存住了选择项<br />
<select id="select1" class="userData">
<option>选项一</option>
<option>选项二</option>
<option>选项三</option>
<option>选项四</option>
</select>
<script>
    var obj=document.all.select1;                //获取页面中的下拉框
    obj.attachEvent('onchange',saveSelectedIndex) //为下拉框绑定更改事件
    function saveSelectedIndex(){                 //保存数据到 userData 中的方法
        obj.setAttribute("sSelectValue",obj.selectedIndex);
        obj.save("oSltIndex");
    }
    window.attachEvent('onload',loadSelectedIndex) //加载数据的事件
    function loadSelectedIndex(){                 //从 userData 中获取数据的方法
        obj.load("oSltIndex");
        obj.selectedIndex=obj.getAttribute("sSelectValue");
    }
</script>
</body>
</html>
```

【运行效果】

重新启用页面后的效果如图 21-2 所示。

【难点剖析】

userData 通常被称为是一种保存数据的行为。使用前必须在 style 元素内添加一行代码 “userData {behavior:url(#default#userData);}”。然后指定要使用 userData 功能的元素的 “class” 属性。

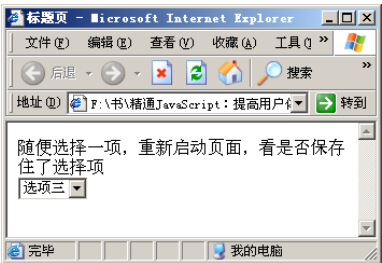


图 21-2 重新启用页面后的效果

21.3 使用 userData 保存 checkbox 标记

【实例描述】

复选框 checkbox 一般用来选择多项数据。如果当用户选择完后关闭了浏览器，重新打开后如何还能正确显示复选框的选择呢？本例学习使用 userData 保存复选框的选择标记。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<style>
    .userData {behavior:url(#default#userData);}
</style>
</head>
<body>
<input type=checkbox id=myChk class=userData>这是使用了 UserData 的复选框

    <script language="javascript">
        var obj=document.all.myChk;                //获取复选框
        obj.attachEvent('onclick',saveChecked)       //为复选框绑定单击事件
        function saveChecked(){
            obj.setAttribute("bCheckedValue",obj.checked); //单击后保存复选框的选中状态
            obj.save("oChkValue");                    //保存在指定名称的存储区域
        }
        window.attachEvent('onload',loadChecked)     //绑定加载事件
        function loadChecked(){
            obj.load("oChkValue");                    //找到指定名称的存储区域
            var chk=(obj.getAttribute("bCheckedValue")==true)?true:false;
                                                    //根据变量值，设置 checkbox 的选中状态
            obj.checked=chk;
        }
    </script>
    <br /><input type=checkbox id=Checkbox1>这是没使用 UserData 的复选框
</body>
</html>
```

【运行效果】

运行页面后选中两个复选框，然后关闭页面，再重新启动页面后的效果如图 21-3 所示。

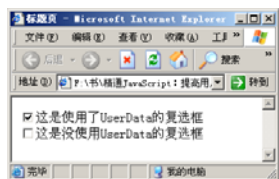


图 21-3 重新启动页面后的效果

【难点剖析】

本例中使用“attachEvent”方法动态地为复选框添加了“onclick”和“onload”事件。当用户选择复选框时，会自动保存用户的选择。这样重新启动页面会触发“onload”事件，并从 userData 的数据存储区域找到复选框的选中状态。

21.4 使用 DOM 实现控件的替换

【实例描述】

本例实现两个 div 内元素互换，主要依靠 DOM 的“replaceChild”方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script type="text/javascript">
function test()
{
    var mydom1=document.getElementById("divlist1");    //获取指定 ID 的 DOM 对象
    var mydom2=document.getElementById("divlist2");    //获取指定 ID 的 DOM 对象
    mydom1.replaceChild(mydom2.firstChild,mydom1.firstChild);
}
</script>
</head>
<body>
<form id="form1" runat="server">
    <input id="btn1" type="button" value="test" onclick="test()"/>
    <div id="divlist1"><p>this is test1</p></div>
    <div id="divlist2"><p>this is test2</p></div>
</form>
</body>
</html>
```

【运行效果】

初始运行效果如图 21-4 所示。替换后的效果如图 21-5 所示。

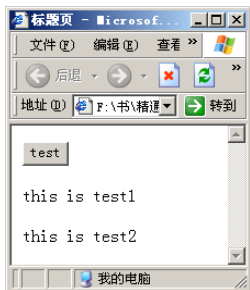


图 21-4 初始运行效果



图 21-5 替换后的效果

【难点剖析】

替换节点需要使用 DOM 中的“replaceChild”方法。需要注意的是此代码的运行结果是“mydom1”中的元素被替换了，而“mydom2”的元素就没有了，“replaceChild”方法相当于转移。

21.5 使用 DOM 实现控件的复制

【实例描述】

复制节点的目的就是实现 div 的拖曳效果，将内容相同的 DOM 对象移动到其他控件中。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script type="text/javascript">
function test()
{
    var mydom1=document.getElementById("divlist1");    //获取指定 ID 的 DOM 对象
    var mydom2=document.getElementById("divlist2");    //获取指定 ID 的 DOM 对象
    mydom2.appendChild(mydom1.childNodes[0].cloneNode(true));
}
</script>
</head>
<body>
<form id="form1" runat="server">
    <input id="btn1" type="button" value="test" onclick="test()"/>
    <div id="divlist1"><p>this is test1</p></div>
    <div id="divlist2"></div>
</form>
</body>
</html>
```

【运行效果】

初始运行效果如图 21-6 所示。复制后的效果如图 21-7 所示。



图 21-6 初始运行效果



图 21-7 复制后的效果

【难点剖析】

本例主要学习点还是 JavaScript DOM 对象的一些操作方法。用“appendChild”方法实现动态添加元素，用“cloneNode”方法实现元素的复制。

21.6 使用 DOM 判断页面中控件是否嵌套

【实例描述】

判断页面控件是否嵌套，主要是判断某元素是否包含子元素。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script type="text/javascript">
function testa()
{
    var mydom=document.getElementById("divtest");    //获取指定 ID 的 DOM 对象
    if(mydom.hasChildNodes())
        alert("有子节点");
    else
        alert("没有子节点");
}
</script>

</head>
<body>
<form id="form1" runat="server">
    <input type="button" value="test" onclick="testa()" />
    <div id="divtest"></div>
</form>
</body>
</html>
```

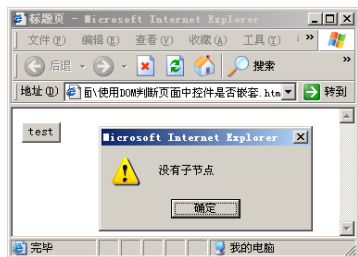


图 21-8 运行提示效果

【运行效果】

运行提示效果如图 21-8 所示。

【难点剖析】

本例首先获取被判断元素的 DOM 对象，这通过“getElementById”方法实现，然后使用 DOM 提供的“hasChildNodes”方法判断是否有子节点。

21.7 使用 DOM 获取页面中某控件的属性

【实例描述】

每个控件都有很多的属性，如 id、name、样式、高度、宽度、value 等。项目中经常需要

修改这些属性的值，本例学习如何获取页面元素的属性。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script type="text/javascript">
function testa()
{
    var mydom=document.getElementById("btn1");    //获取指定 ID 的 DOM 对象
    var myattri=mydom.getAttribute("type");        //获取元素的类型
    alert(" 按钮的英文类型是: "+myattri);
}
</script>

</head>
<body>
    <form id="form1" runat="server">
        <input id="btn1" type="button" value="test" onclick="testa()" />
    </form>
</body>
</html>
```

【运行效果】

运行的判断效果如图 21-9 所示。

【难点剖析】

要获取某元素的属性,首先要创建包含这个元素的 DOM 对象,然后使用“getAttribute”方法获取此元素指定的属性并显示出来。

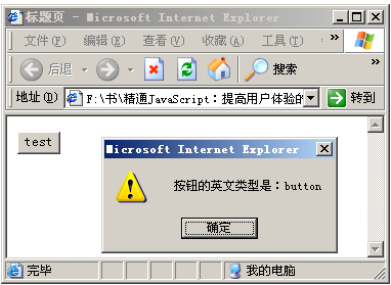


图 21-9 运行的判断效果

21.8 将某行排在表格的最后

【实例描述】

表格的行可以任意移动，本例学习如何将某行移动到最后一行，或者移动到第一行。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
function moveToLast(tr)
{
    var tb =document.getElementById(tr).parentElement.parentElement;//获取表格对象
    tb.moveRow(document.getElementById(tr).RowIndex, -1);//移动表格中的行到指定位置
}
</script>
</head>
<body>
```

```
</script>
</head>
<body>
<table border="1">
<tr id="tr1"><td>第一行</td></tr>
<tr id="tr2"><td>第二行</td></tr>
<tr id="tr3"><td>第三行</td></tr>
</table>
<br>
<input type="button" value="把第一行排到最后" onclick="moveToLast('tr1')" />
</body>
</html>
```

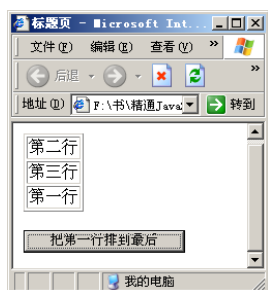


图 21-10 移动第一行到最后一行后的效果

【运行效果】

移动第一行到最后一行后的效果如图 21-10 所示。

【难点剖析】

本例使用“parentElement”属性获取表格对象，然后利用表格对象的“moveRow”方法将指定的行移动到指定的位置。参数“-1”表示将指定的行移动到最后，如果是“0”，则表示移动到第一行。

21.9 动态删除页面中的元素

【实例描述】

可以使用 DOM 删除页面中指定的元素而不刷新页面。本例介绍如何实现动态删除。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script type="text/javascript">
function test()
{
    var mydom=document.getElementById("divlist1"); //获取指定 ID 的 DOM 对象
    var childnode=mydom.firstChild; //获取被删除的节点
    mydom.removeChild(childnode); //删除节点
}
</script>
</head>
<body>
<form id="form1" runat="server">
    <input id="btn1" type="button" value="test" onclick="test()"/>
    <div id="divlist1"><p>this is test1</p></div>
</form>
```



```
</body>
</html>
```

【运行效果】

初始运行效果如图 21-11 所示。删除后的效果如图 21-12 所示。



图 21-11 初始运行效果



图 21-12 删除后的效果

【难点剖析】

实现删除需要指明两个节点，一个是要删除的节点，一个是被删除节点的父节点。使用“removeChild”方法可以实现删除。

21.10 克隆表格

【实例描述】

表格的行可以插入、删除和修改，类似于 C# 中的 DataTable 对象。大多数的表格对象都支持克隆功能，本例就学习如何克隆 HTML 的 table。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<input type=button value=克隆表格 name=mytxt onclick=add()>
<table border=1>
<tr id=r1 name=r1>
    <td>第一列</td><td>第二列</td>
</tr>
</table>
<script language=JavaScript>
i=1;
function add()
{
    //用于表格 id 的自增长
    ++i;
```

```
var newTr= r1.cloneNode(true);           //克隆指定的行
newTr.id="r"+i;                           //指定新行的 id
newTr.name="r"+i;                         //指定新行的 name
r1.parentNode.insertBefore("beforeEnd",newTr); //在当前表格中插入行
}
</script>
</body>
</html>
```

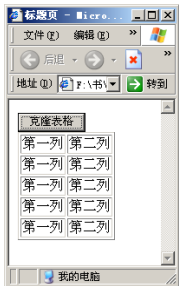


图 21-13 克隆多个表格的效果

【运行效果】
克隆多个表格的效果如图 21-13 所示。

【难点剖析】
本例的重点有两个，如何克隆行，如何将克隆的行添加到当前表格。克隆行使用了行的“cloneNode”方法，其只有一个参数，指明被克隆的节点是否应该包含所有的属性及原始节点中的子节点。将克隆行添加到当前表格的方法是“insertAdjacentElement”，其包含两个参数，要插入的位置和要插入的行。

21.11 拖动表格行改变顺序

【实例描述】
表格的行可以通过后台命令实现互换，也可以让用户自己使用拖曳的方式完成互换操作。本例学习一个简单的拖曳表格行操作。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>拖曳表格的行</title>
<style type="text/css">
td{position:relative;
}
body {
margin-left: 1px;
margin-top: 1px;
margin-right: 1px;
margin-bottom: 1px;
}
</style>
</head>

<body style="margin-left:0px; margin-top:0px;">
```

```

<table width="281" border="1" bordercolor="#CCCCCC" id="dragTbl">
<tr>
  <td width="73">序号</td>
  <td width="52">姓名</td>
  <td width="65">年龄</td>
  <td width="63">地址</td>
</tr>
<tr>
  <td width="73">1</td>
  <td width="52">张三</td>
  <td width="65">22</td>
  <td width="63">北京海淀</td>
</tr>
<tr>
  <td >2</td>
  <td>王五</td>
  <td>24</td>
  <td>上海浦东</td>
</tr>
<tr>
  <td>3</td>
  <td>朝气</td>
  <td>24</td>
  <td>广州深圳</td>
</tr>
</table>
<script language="javascript" >
var obj;
var xx=0,yy=0;
var tagobj;
var dragobj;
function draginit(){
  var tblRows = document.getElementsByTagName("TR"); //获取所有的表格行
  for(var i=0;i<tblRows.length;i++){ //遍历每一行
    if((tblRows[i].parentNode.parentNode.id).toString().indexOf("drag")!=-1){
      tblRows[i].onmousedown=mousedown; //绑定所有的鼠标事件
      tblRows[i].ondragover=dragover;
      tblRows[i].ondragend=dragend;
      tblRows[i].ondrag=dragmove;
      tblRows[i].style.position="relative";
      tblRows[i].style.zIndex=1;
    }
  }
}
function mousedown(){ //鼠标按下时的处理
  obj = event.srcElement;
  if(obj.tagName=="TD") obj=obj.parentNode; //如果是单元格

```



```
if(obj.tagName!="TR") return false; //如果是单元行
if(obj.rowIndex==0) return false; //如果是标题
yy=event.clientY; //鼠标的 x 坐标和 y 坐标
xx=event.clientX;
obj.style.zIndex=0;
try{
    obj.dragDrop();
}catch(e){
}
}
function dragmove(){ //表格拖曳时的位置获取
    obj.style.top = event.clientY-yy;
    obj.style.left = event.clientX-xx;
}
function dragover(){ //鼠标拖动时的操作
    tagobj=event.srcElement;
    if(tagobj.tagName=="TD"){tagobj=tagobj.parentNode;} //如果是单元格
    if(tagobj.tagName!="TR")return false; //如果是单元行
}
function dragend(){ //拖曳完毕后的处理，高度，宽度等
    obj.style.top=0;
    obj.style.left=0;
    obj.style.zIndex=1;
    if(tagobj!=null && tagobj.rowIndex!=0){
        var t1=dragTbl.rows[obj.rowIndex];
        var t2=dragTbl.rows[tagobj.rowIndex];
        dragTbl.getElementsByTagName('tbody')[0].insertBefore(obj,tagobj);
        //实现表格单元行的互换

        tagobj.style.zIndex=1;
    }

    tagobj=null;
}
draginit(); //初始化拖曳操作
</script>
</body>
</html>
```

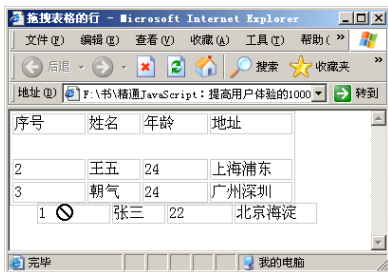


图 21-14 拖曳表格行的效果

【运行效果】

拖曳表格行的效果如图 21-14 所示。

【难点剖析】

本例的重点是为表格绑定的事件，其中每个事件都对应一个自定义的方法。难点是根据鼠标的坐标改变表格行的位置。最终实现互换的原理是使用“insertBefore”方法将两个行的位置互换。

21.12 表格操作常用方法

【实例描述】

很多开发工具提供了高性能的表格控件，可以实现行的添加、删除、移动、选择等。本例将使用 JavaScript 实现 HTML table 的这些功能。

【实现代码】

```
Function.prototype.bind = function() { //数据的绑定
    var __method = this, args = $A(arguments), object = args.shift();
    return function() {
        return __method.apply(object, args.concat($A(arguments)));
    }
}
function CTable(id,rows){
    this.tbl=typeof(id)=="string"?document.getElementById(id):id;
    if (rows && /\d+$/ .test(rows)) this.addrows(rows) //为表格添加行数
}
CTable.prototype={
    addrows:function(n){ //随机添加 n 行
        new Array(n).each(this.add.bind(this))
    },
    add:function(){ //添加一行
        var self=this;
        var tr = self.tbl.insertRow(-1),td1= tr.insertCell(-1),td2= tr.insertCell(-1), td3=
tr.insertCell(-1);
        var chkbox=document.createElement("INPUT")
        chkbox.type="checkbox"
        chkbox.onclick=self.highlight.bind(self)
        td1.appendChild(chkbox) //第一列添加复选框
        td1.setAttribute("width","35")
        td2.innerHTML=Math.ceil(Math.random()*99) //第二列的随机填充值
        td3.innerHTML=Math.ceil(Math.random()*99) //第三列的随机填充值
    },
    del:function(){ //删除所选行
        var self=this
        $A(self.tbl.rows).each(function(tr){if (self.getChkBox(tr).checked)tr.parentNode.
removeChild(tr)})
    }
}
```

由于篇幅所限，本例只截取了几个重要的方法，详细的代码可参考随书光盘。

【运行效果】

表格选择行的效果如图 21-15 所示。

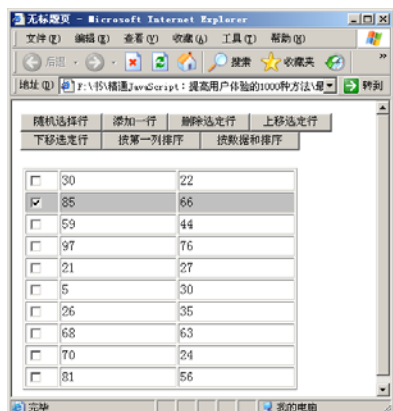


图 21-15 表格选择行的效果

【难点剖析】

本例的重点是如何使用 JavaScript 创建表格类，并在类中设计表格的常用方法。在本例中手动创建了“add”、“del”等表格操作方法，读者可通过代码注释了解具体的代码实现。

第22章

流行应用：Ajax 和 Property 的技巧

本章导读

一直以来，JavaScript 扮演着客户端脚本的角色，很多人认为其不属于面向对象的开发语言，但随着 Property.js 框架的兴起，开发人员可以感受到 JavaScript 强大的面向对象能力。Ajax 是最新的 JavaScript 应用，可以异步传输数据，提高了页面的加载速度。本章将介绍 Property 和 Ajax 的常用技巧。



22.1 关机特效（一）

【实例描述】

有些网页中的功能使用都需先决条件，如只有注册用户才能访问页面，如果没有注册则不能访问。可以通过模式窗口实现登录功能，也可以使用本例介绍的关机效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
var timer                                //定时器
var Opacity = 0                          //背景覆盖色的透明度
function hide() {
    myDiv.style.display = "block"         //显示div层
    myDiv.style.height = document.body.scrollHeight
                                           //要设置div的高度同窗体相同,以实现覆盖
    Opacity = 0
    event.srcElement.blur()              //当前对象失去焦点
    timer = window.setInterval("AddOpacity()", 5)
                                           //使用定时器逐渐增加窗体的透明度
}
function AddOpacity()                    //变更透明度
{
    if (Opacity == 50)
    {
        window.clearInterval(timer)      //清空定时器
        myBtn.style.display = "block"     //显示按钮
        return
    }
    Opacity = ( Opacity + 3 ) > 50 ? 50 : Opacity + 3 //判断透明度
    myDiv.style.filter = "Alpha(Opacity=" + Opacity + ")" //设置层的样式
}
function increOpa()                      //透明度减小
{
    if (Opacity == 0)
    {
        window.clearInterval(timer)      //清空定时器
        myDiv.style.display = "none"      //屏蔽层的显示
        return
    }
    Opacity = (Opacity - 3 < 0) ? 0 : Opacity - 3 //变更透明度
    myDiv.style.filter = "Alpha(Opacity=" + Opacity + ")" //设置层的样式
}
function btnChange()
```



```
{
    timer = window.setInterval('increOpa()', 5);           //设置定时器
    myBtn.style.display='none';                             //屏蔽按钮的显示
}
</script>
</head>
<body topmargin=0 leftmargin=0>
    <div id=myDiv style="position:absolute;z-index:99; background-color:darkgray;
Filter: Alpha(Opacity=0); border:1px solid
#333333;display:none;width:100%;vertical-align:center;text-align:center">&nbsp;</div>
    <button onclick=hide()>hide</button>
    <button id=myBtn style="position:absolute; left:100;top:200;z-index:100;display:none"
onclick="btnChange()">返回</button>
    <script>
        for (var i=0; i<30; i++)
            document.write ("<p>这是主要的页面，看看能不能动</p>")           //设置页面的内容，最好能超//
过一页，以看到屏蔽效果
    </script>
</body>
</html>
```

【运行效果】

关机效果的界面如图 22-1 所示。

【难点剖析】

本例的重点其实是 CSS 样式表，JavaScript 的作用是实现样式表的动态改变。filter 是样式表的滤镜，可以实现一些很炫的页面效果（如文字发光）。本例中使用了“Alpha(Opacity)”透明度属性。

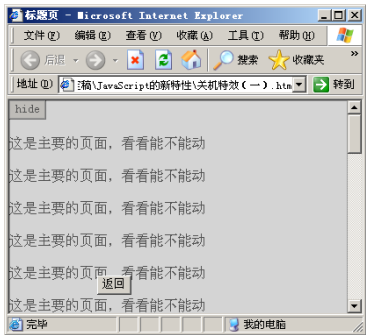


图 22-1 关机效果的界面

22.2 关机特效（二）

【实例描述】

本例应该算是最简单的关机特效。当用户单击“关机效果”链接时模拟 Windows 操作系统关机的效果。

【实现代码】

```
<html>
<head>
<title>关机特效</title>
</head>
<body style="background-color:#E9EDF7">
<script language="javascript">
function closeMsg(txt)
{
```

```
document.body.style.filter='alpha(opacity=40) gray'; //实现窗体的滤镜效果，设置透明度
confirm(txt); //弹出提示性对话框
document.body.style.filter=''; //取消滤镜
}
setTimeout("msgbox('注销 关闭 重新启动')",1000) //隔一秒触发一次
</script>
<a href="javascript:closeMsg('注销 关闭 重新启动')">关机效果</a>
</body>
</html>
```

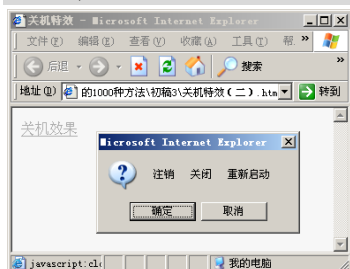


图 22-2 单击“关机效果”链接后的效果

【运行效果】

单击“关机效果”链接后的效果如图 22-2 所示。

【难点剖析】

本例中通过设置 body 标签的滤镜效果实现了灰色层的覆盖。使用“confirm”方法弹出一个类似模式的对话框，用户只有选择“确定”或“取消”按钮后，才可以执行其他的操作。

22.3 评星效果

【实例描述】

评星效果在很多网站起到调查的作用，如图书网站上读者对新书的评价。本例学习如何制作这种评星效果。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script>
var starNum=0;
function chgStar()
{
    starNum=event.srcElement.id.slice(-1); //从第一个到结束
    for (var i=1;i<=6;i++) eval("id"+i).innerText="☆"; //显示的星星
    for (var j=1;j<=starNum;j++) eval("id"+j).innerText="★"; //选中的星星
}

function getStar()
{
    alert("用户的评价是"+starNum+"颗星!"); //显示评级数
}

for (var i=1;i<=6;i++)
{
    //动态输出 span，并设置 span 的样式和事件
```

```
document.write('<span id="id'+i+' onclick="chgStar()" style="cursor: hand;">☆  
</span>');  
}  
document.write('<br/><br/><button onclick="getStar()">评级分数</button>');  
//动态输出按钮，包括其事件  
  
</script>  
</head>  
<body>  
</body>  
</html>
```

【运行效果】

本例的运行效果如图 22-3 所示。

【难点剖析】

本例的难点是用户选中第 5 颗星时，此星前面的星都要变成黑色。“starNum”变量获取的是用户选择的星星的“id”，使用“for (var j=1;j<=starNum;j++)”语句循环设置对应的“span”元素的内容为黑色星星。

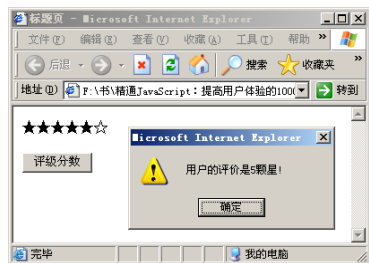


图 22-3 本例的运行效果

22.4 输入框自动完成功能

【实例描述】

输入框自动完成是目前比较热门的一种技术，可利用 Ajax 技术实现无刷新提示功能。本例介绍这种技巧的原理。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<title>标题页</title>  
</head>  
<body>  
<input onkeyup="showtips();if(event.keyCode==27)hideDiv();" id=txt onkeydown=  
'enterTips()'>(如: ShanDong)<br/>  
<select id=sel style='display:none' onclick=viewTxt() onkeydown='if(event.keyCode=  
=13)viewTxt()'></select>  
<script>  
var city = new Array("Guangdong","Tianjing","Shanghai","Beijing","ShanDong",  
"Shanxi","Hunan","shangdi");  
var city2=new Array("广东","天津","上海","北京","山东","陕西","湖南","上地");  
function showtips(){  
    obj=event.srcElement; //获取操作对象  
    sel.length=0; //列表框的长度  
    var len=city.length; //数组的长度  
    var re=new RegExp("^"+obj.value,"i") //正则表达式——搜索用户输入的值
```



```
var j=0
for(i=0;i<len;i++){
    if(re.test(city[i])==true){                                //如果存在搜索的值
        sel.style.display='';                                //显示提示层
        sel.add(new Option(city[i],city2[i]));j++;} //提示信息
    sel.size = (j>1)?j:2;
}

function enterTips(){
    e=event.keyCode;
    if(sel.style.display!='none'){                            //如果提示层没有隐藏
        if(e==13) event.srcElement.value=sel.value,sel.style.display='none'; //回车
        if(e==40) sel.focus();                                //使用下箭头时，提示层获得焦点
    }
}

function viewTxt(){
    txt.value=sel.value;                                       //显示选择的内容
    hideDiv()                                                  //隐藏提示层
}

function hideDiv(){
    sel.style.display='none';                                  //隐藏提示层的显示
    txt.focus()                                                 //文本框获得焦点
}

document.onclick=function(){                                  //单击窗体时，隐藏提示层
    hideDiv()
}

</script>
</body>
</html>
```

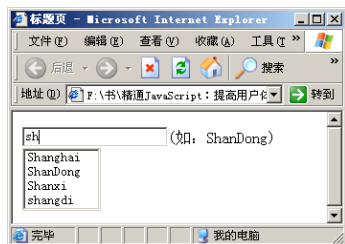


图 22-4 单击单元格后的效果

见，这样就实现了自动提示功能。

【运行效果】

单击单元格后的效果如图 22-4 所示。

【难点剖析】

本例通过一个正则表达式完成了自动提示的功能。首先使用“new RegExp”创建了一个正则表达式“re”，然后使用正则的“test”方法，在现有的文本提示信息中查找匹配正则的数据。使用“sel.add”方法将搜索出的数据添加到列表中，并设置列表框为可

22.5 Ajax 效果的字符串过滤

【实例描述】

在 Microsoft 提供的 Ajax 框架中，有个根据第一个字母选择单词的过滤实例。本例将学习

如何使用正则实现这种效果。

【实现代码】

```
<script type="text/javascript">
function filterlist(selectobj) {
    // 过滤对象
    this.selectobj = selectobj;
    //要过滤的字符, i 表示忽略大小写, ""表示不忽略
    this.flags = 'i';
    // 要过滤的项, 选择是文本还是值
    this.match_text = true;
    this.match_value = false;
    //调试参数
    this.show_debug = false;
    //初始化的方法
    this.init = function() {
        if (!this.selectobj) return this.debug('没有实现过滤的控件');
        if (!this.selectobj.options) return this.debug('过滤控件中没有内容');
        //制作选择项的副本
        this.optionscopy = new Array();
        if (this.selectobj && this.selectobj.options) {
            for (var i=0; i < this.selectobj.options.length; i++) {
                // 创建一个新的选择项对象
                this.optionscopy[i] = new Option();
                // 设置选择项的文本
                this.optionscopy[i].text = selectobj.options[i].text;
                // 设置选择项的值
                if (selectobj.options[i].value) {
                    this.optionscopy[i].value = selectobj.options[i].value;
                } else {
                    this.optionscopy[i].value = selectobj.options[i].text;
                }
            }
        }
    }
    this.reset = function() { //重置选择项
        this.set('');
    }
    //实现过滤的方法
    this.set = function(pattern) {
        var loop=0, index=0, regexp, e;
        if (!this.selectobj) return this.debug('没有实现过滤的控件');
        if (!this.selectobj.options) return this.debug('过滤控件中没有内容');
        // 清空列表中的内容
        this.selectobj.options.length = 0;
        // 使用正则表达式实现字符过滤, 初始化正则表达式
        try {
```



```
        regexp = new RegExp(pattern, this.flags);
    } catch(e) {
        if (typeof this.hook == 'function') {
            this.hook();
        }
        return;
    }
    // 循环添加过滤后的结果
    for (loop=0; loop < this.optionscopy.length; loop++) {
        // 定义选择项
        var option = this.optionscopy[loop];
        // 实现正则表达式过滤
        if ((this.match_text && regexp.test(option.text)) ||
            (this.match_value && regexp.test(option.value))) {
            // 使用过滤结果创建新选择项
            this.selectobj.options[index++] =
                new Option(option.text, option.value, false);
        }
    }
    if (typeof this.hook == 'function') {
        this.hook();
    }
}

//设置正则表达式的过滤标志
this.set_ignore_case = function(value) {
    if (value) {
        this.flags = 'i';
    } else {
        this.flags = '';
    }
}
this.debug = function(msg) { //调试方法
    if (this.show_debug) {
        alert('过滤结果: ' + msg);
    }
}
this.init(); //调用初始化方法
}
</script>
```

【运行效果】

字符串过滤的效果如图 22-5 所示。

【难点剖析】

本例的重点是使用正则表达式实现字符的过滤，还实现了动态添加列表项的功能。代码中使用“new”关键字创建了 RegExp 对象，其用来提供简单的正则表达式支持功能。使用 RegExp 对象的“test”方法，可以对指定的字符串执行一个正则表达式搜索，并返回一个布尔值指示是

否找到匹配的模式。当搜索到结果时就使用“new Option”动态创建一个列表项，同时添加到列表中。

22.6 GMail 右上角的 Loading 效果

【实例描述】

在打开 GMail 邮箱的时候，如果邮箱中内容太多会在页面的右上角出现一个提示等待的效果。本例模拟这个效果，实现一种页面加载特效。

【实现代码】

```
<html>
<head>
<title>加载效果</title>
<script language="javascript">
function hideMsg()
{
    document.getElementById("msgDiv").innerHTML="";           //取消等待状态的显示
}
</script>
</head>
<body onunload="window.open('http://http://www.google.com')" scroll=no topmargin=0
leftmargin=0 rightmargin=0 bottommargin=0>
    <div style="position:absolute;top:10;left:10;" id="msgDiv"><font color=red>
loading.....</font></div>
    <iframe onload="hideMsg()" style="position:absolute;top:40;left:10;" src="http:
//www.sina.com.cn" width=300 height=200></iframe>
</body>
</html>
```

【运行效果】

加载提示效果如图 22-6 所示。

【难点剖析】

本例使用一个 div 来显示加载过程中的提示。然后使用 iframe 框架来加载一个页面，“onload”事件调用加载完毕后要执行的操作。加载完成后就取消显示 div 层的加载提示文本，此处直接隐藏 div 也可以。



图 22-5 字符串过滤的效果

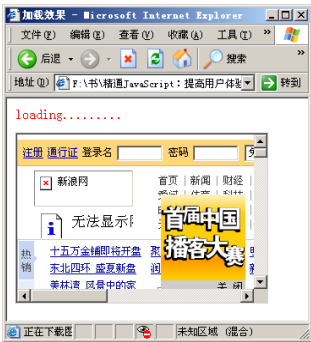


图 22-6 加载提示效果

22.7 使用 XMLHTTP 获取天气预报

【实例描述】

使用 XMLHTTP 可以访问网络上其他页面，并获取页面的源代码。本例通过 XMLHTTP 获



取天气预报网页的内容并显示在当前窗体中。

【实现代码】

```
<HTML>
<HEAD>
<TITLE>天气预报</TITLE>
<script language="javascript">
    var xmlhttp;
    function getWeather()
    {
        //创建异步对象
        xmlhttp=new ActiveXObject("Msxml2.XMLHTTP");
        //加载服务器,注意 URL 参数的使用
        xmlhttp.Open("GET","http://tw.weather.yahoo.com/world_single.html? city=
10101",false)
        //异步对象事件挂钩
        xmlhttp.onreadystatechange=stateChange;
        //发送请求,无参数
        xmlhttp.Send(null);
    }
    function stateChange()
    {
        if(xmlhttp.readyState==4 && xmlhttp.status==200)
        {
            //获取所有返回的数据
            var data=xmlhttp.responseText;
            //过滤自己需要的数据
            var begin=data.indexOf("国际个别都市 start");
            var end=data.indexOf("国际个别都市 end");
            var weather=data.substring(begin+15,end);
            //填充天气内容
            document.getElementById("divweather").innerHTML=weather;
            //显示结果
            document.getElementById("divweather").style.visibility="visible";
        }
    }
</script>
</HEAD>
<BODY onload="getWeather()">
<div align="center" id="today_time">今天的日期
</div>
<div align="center" id="divweather"></div>
<script language="javascript">
//设置显示星期几,用数组存储
var x = new Array("星期日", "星期一", "星期二");
var x = x.concat("星期三", "星期四", "星期五");
var x = x.concat("星期六");
```



```
var today_time = new Date(); //获取当天的日期
//显示用中文表示的日期
document.all("today_time").innerText=today_time.getFullYear()+'年'+(today_ time.
getMonth()+1)+'月'+today_time.getDate()+'日'+x[today_time.getDay()];
</script>
</BODY>
</HTML>
```

【运行效果】

获取天气预报的效果如图 22-7 所示。



图 22-7 获取天气预报的效果

【难点剖析】

本例的难点是如何使用 XMLHTTP。使用 XMLHTTP 具体步骤如下：

- (1) 创建 XMLHTTP 对象，使用语句 “new ActiveXObject("Msxml2.XMLHTTP")”；
- (2) 打开要访问的网页，使用 “xmlhttp.Open” 方法；
- (3) 向指定网页发送数据，使用 “xmlhttp.Send” 方法；
- (4) 获取网页的数据，使用 “xmlhttp.responseText” 属性。

22.8 拖曳任意对象

【实例描述】

网页中可能经常遇到需要拖曳 div 或 table 的情况。本例学习一种方法，可实现对任意对象的拖曳。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
```



```
function DragEvent()  
{  
    //参数  
    // 开始时的鼠标在对象中的偏移位置  
    //DragFlag 0:拖曳停止 1:拖曳开始  
    this.x = 0;  
    this.y = 0;  
    this.DragFlag=0;  
}  
var DragObject = new DragEvent();  
  
function DragMoveObject(obj)  
{  
    if(event.button == 1) //如果按下的是鼠标左键  
    {  
        obj.style.position="absolute"; //设置对象为绝对定位模式  
        if(DragObject.DragFlag==0) //拖曳开始  
        {  
            DragObject.DragFlag = 1;  
            DragObject.x = event.offsetX; //鼠标的 x 坐标  
            DragObject.y = event.offsetY; //鼠标的 y 坐标  
        }  
        obj.style.left = event.x-DragObject.x; //保持鼠标在对象中的位置不变  
        obj.style.top = event.y-DragObject.y;  
    }  
    else  
    {  
        DragObject.DragFlag = 0; //拖曳停止  
    }  
}  
</script>  
</head>  
<body>  
<textarea cols="30" rows="5" onmousedown="DragMoveObject(this);" ></textarea><br />  
<input type=button value="拖曳" onmousedown="DragMoveObject(this);" />  
<input id="Button1" type="button" value="button"onmousedown="DragMoveObject (this);" />  
</body>  
</html>
```

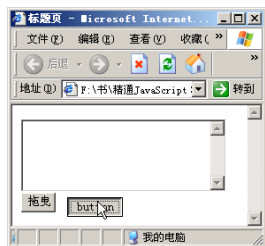


图 22-8 本例的拖曳效果

【运行效果】

本例的拖曳效果如图 22-8 所示。由于没有设置鼠标的形状，所以效果并不明显。

【难点剖析】

本例的重点是捕获鼠标的移动坐标，然后设置指定标签的坐标跟随鼠标的坐标。判断是否按下鼠标使用的条件是“event.button = 1”。

22.9 避免打开无效页面

【实例描述】

用户访问页面时，由于页面或服务器的的问题，可能会提示错误或根本打不开网页，为了避免让用户看到这些错误，可使用 XMLHTTP 对象实现提前判断。

【实现代码】

```
<script language="javascript">
function getURL(url)
{
    var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");           //创建 XMLHTTP 对象
    xmlhttp.open("GET",url,false);                                   //打开用户指定的导航页
    xmlhttp.send();                                                  //发送信息
    if (xmlhttp.readyState==4 && xmlhttp.Status==200)
        return false;                                              //返回 false 表示发送不成功
    }
function test(e, url)
{
    if(!getURL(e.href))
        e.href = url;                                              //转换导航链接
    }
}</script>
```

需要在 body 中添加一个链接，并调用“test”方法，代码如下所示：

```
<a href="http://www.google.com" onclick="test(this, 'http://www.baidu.com/')">打开测试页</a>
```

【运行效果】

转换成功后的界面如图 22-9 所示，本例中假设访问 Google 发生错误时导航到百度搜索。

【难点剖析】

本例中的 XMLHTTP 对象是实现此功能的关键，用其创建对指定页面的访问，并通过“xmlhttp.readyState”和“xmlhttp.Status”来获取访问的返回状态，如果“if (xmlhttp.readyState==4 && xmlhttp.Status==200)”条件满足，则返回“true”表示访问成功，本例为了测试运行效果，此处返回了“false”。



图 22-9 转换成功后的界面

22.10 用 JavaScript 调用 Google AdSense

【实例描述】

Google AdSense 是 Google 为网站提供广告的服务。本例学习如何使用 JavaScript 调用这个

广告服务，并在自己的网站中显示这些广告。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>中文搜索</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<div>
<script type="text/javascript">
//以下是 Google AdSense 服务的参数
google_ad_client = "pub-5660892815235048";
google_ad_width = 336;
google_ad_height = 280;
google_ad_format = "336x280_as";
google_ad_type = "text_image";
google_ad_channel = "";
google_color_border = "FFFD7F";
google_color_bg = "FFFD7F";
google_color_link = "0000FF";
google_color_text = "191919";
google_color_url = "7F7F7F";
</script>
<script type="text/javascript"
  src="http://pagead2.googlesyndication.com/pagead/show_ads.js">
//此处是服务的 js 代码
</script>
</div>
</body>
</html>
```



图 22-10 调用广告效果

【运行效果】

调用广告效果如图 22-10 所示。

【难点剖析】

本例的重点是 Google AdSense 参数的调用，以及显示广告内容的 JavaScript 文件“show_ ads.js”。调用方法不能随意，具体解释可参考 Google AdSense 的官方网页。

22.11 Ajax 效果——可拖曳的表格

【实例描述】

Ajax 技术已经被越来越多的人关注，其允许用户自己定制界面。本例介绍如何利用 Ajax 实现可拖曳的表格，允许用户通过拖曳表格定制自己的界面。

【实现代码】

```

<script defer>
var Drag={dragged:false,ao:null,tdiv:null,dragStart:function()
{ //创建新的DIV
    Drag.ao=event.srcElement;
    if((Drag.ao.tagName=="TD")||(Drag.ao.tagName=="TR")){
        Drag.ao=Drag.ao.offsetParent;
        Drag.ao.style.zIndex=100;
    }else
        return;
    Drag.dragged=true;
    Drag.tdiv=document.createElement("div");
    Drag.tdiv.innerHTML=Drag.ao.outerHTML;
    Drag.ao.style.border="1px dashed red";
    Drag.tdiv.style.display="block";
    Drag.tdiv.style.position="absolute";
    Drag.tdiv.style.filter="alpha(opacity=70)";
    Drag.tdiv.style.cursor="move";
    Drag.tdiv.style.border="1px solid #000000";
    Drag.tdiv.style.width=Drag.ao.offsetWidth;
    Drag.tdiv.style.height=Drag.ao.offsetHeight;
    Drag.tdiv.style.top=Drag.getInfo(Drag.ao).top;
    Drag.tdiv.style.left=Drag.getInfo(Drag.ao).left;
    document.body.appendChild(Drag.tdiv);
    Drag.lastX=event.clientX;
    Drag.lastY=event.clientY;
    Drag.lastLeft=Drag.tdiv.style.left;
    Drag.lastTop=Drag.tdiv.style.top;
},
draging:function(){ //判断鼠标的位置
    if(!Drag.dragged||Drag.ao==null)return;
    var tX=event.clientX;
    var tY=event.clientY;
    Drag.tdiv.style.left=parseInt(Drag.lastLeft)+tX-Drag.lastX;
    Drag.tdiv.style.top=parseInt(Drag.lastTop)+tY-Drag.lastY;
    for(var i=0;i<parentTable.cells.length;i++){
        var parentCell=Drag.getInfo(parentTable.cells[i]);
        if(tX>parentCell.left&&tX<=parentCell.right&&tY>=parentCell.top&&tY<=
parentCell.bottom){
            var subTables=parentTable.cells[i].getElementsByTagName("table");
            if(subTables.length==0){
                if(tX>parentCell.left&&tX<=parentCell.right&&tY>=parentCell.top&&tY<=
parentCell.bottom){
                    parentTable.cells[i].appendChild(Drag.ao);
                }
                break;
            }
        }
    }
}

```



```

        for(var j=0;j<subTables.length;j++){
            var subTable=Drag.getInfo(subTables[j]);
            if(tX>=subTable.left&&tX<=subTable.right&&tY>=subTable.top&&tY<=
subTable.bottom){
                parentTable.cells[i].insertBefore(Drag.ao,subTables[j]);
                break;
            }else{
                parentTable.cells[i].appendChild(Drag.ao);
            }
        }
    }
},
dragEnd:function(){ //拖曳完毕
    if(!Drag.dragged)return;
    Drag.dragged=false;
    Drag.mm=Drag.repos(150,15);
    Drag.ao.style.borderWidth="0px";
    Drag.ao.style.borderTop="1px solid #3366cc";
    Drag.tdiv.style.borderWidth="0px";
    Drag.ao.style.zIndex=1;
},
getInfo:function(o){ //取得坐标
    var to=new Object();
    to.left=to.right=to.top=to.bottom=0;
    var twidth=o.offsetWidth;
    var theight=o.offsetHeight;
    while(o!=document.body){
        to.left+=o.offsetLeft;
        to.top+=o.offsetTop;
        o=o.offsetParent;
    }
    to.right=to.left+twidth;
    to.bottom=to.top+theight;
    return to;
},
repos:function(aa,ab){
    var f=Drag.tdiv.filters.alpha.opacity;
    var tl=parseInt(Drag.getInfo(Drag.tdiv).left);
    var tt=parseInt(Drag.getInfo(Drag.tdiv).top);
    var kl=(tl-Drag.getInfo(Drag.ao).left)/ab;
    var kt=(tt-Drag.getInfo(Drag.ao).top)/ab;
    var kf=f/ab;
    return setInterval(function(){if(ab<1){
        clearInterval(Drag.mm);
        Drag.tdiv.removeNode(true);
        Drag.ao=null;
        return;
    }

```

```

        }
        ab--;
        tl-=kl;
        tt-=kt;
        f-=kf;
        Drag.tdiv.style.left=parseInt(tl)+"px";
        Drag.tdiv.style.top=parseInt(tt)+"px";
        Drag.tdiv.filters.alpha.opacity=f;
    },aa/ab)
},
inint:function(){ //初始化表格
    for(var i=0;i<parentTable.cells.length;i++){
        var subTables=parentTable.cells[i].getElementsByTagName("table");
        for(var j=0;j<subTables.length;j++){
            if(subTables[j].className!="dragTable")break;
            subTables[j].rows[0].className="dragTR";
            subTables[j].rows[0].attachEvent("onmousedown",Drag.dragStart);
        }
    }
    document.onmousemove=Drag.dragging;
    document.onmouseup=Drag.dragEnd;
}
}
Drag.inint();
</script>

```

拖曳标题栏需要专门的样式。本例主要用到的样式表如下所示：

```

<style>
*{font-size:12px}
.dragTable{
    font-size:12px;
    border-top:1px solid #3366cc;
    margin-bottom: 10px;
    width:100%;
    background-color:#FFFFFF;
}
td{vertical-align:top;}
.dragTR{
    cursor:move;
    color:#7787cc;
    background-color:#e5eef9;
    height:20px;
    padding-left:5px;
    font-weight:bold;
}
#parentTable{
    border-collapse:collapse;

```



```
letter-spacing:25px;  
}  
</style>
```

需要在 body 中添加 table，由于代码内容比较多，此处未给出，可从随书光盘中获取。

【运行效果】

表格的运行效果如图 22-11 所示。表格的拖曳效果如图 22-12 所示。

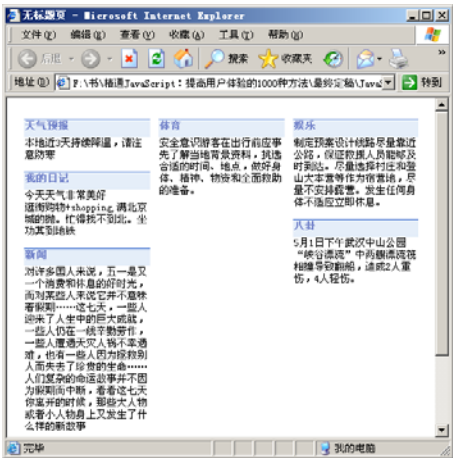


图 22-11 表格的运行效果

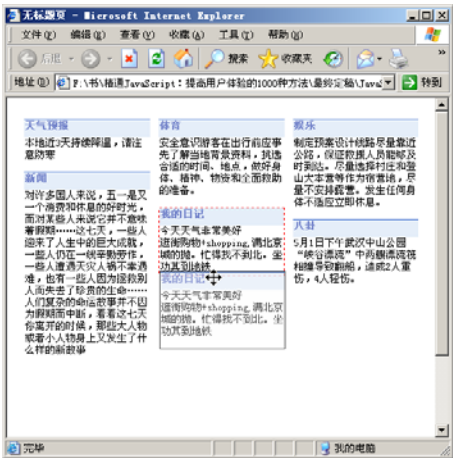


图 22-12 表格的拖曳效果

【难点剖析】

本例中 script 元素后面不再是“language=’javascript’”的属性，而使用了“defer”。“defer”表示在页面加载完后再运行 JavaScript 代码，这样可以防止出现找不到对象的问题。“defer”还可以使脚本在后台被下载，前台的内容则正常显示给用户。

22.12 JavaScript 调用 Web Service

【实例描述】

Web Service 也叫 Web 服务，是目前很流行的一种网络代码共享手段。可以在自己的网站中调用其他人提供的服务，达到自己网站需要的效果（如在自己的网页中添加天气预报服务）。

【实现代码】

创建服务的代码如下所示（此服务需要在 Visual Studio 2005 中创建）。

```
using System;  
using System.Web;  
using System.Collections;  
using System.Web.Services;  
using System.Web.Services.Protocols;
```



```
// <summary>
// 返回 HelloWorld
// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class HelloWorld : System.Web.Services.WebService
{
    [WebMethod]
    public int Hello_World(int x,int y) {
        return x * y;
    }
}
```

用 JavaScript 调用服务的代码如下所示：

```
<HTML>
<HEAD>
    <title>Javascript</title>
    <meta name="GENERATOR" Content="Microsoft Visual Studio .Net 7.1">
    <meta name="CODE_LANGUAGE" Content="C#">
    <meta name="vs_defaultClientScript" content="JavaScript">
    <meta name="vs_targetSchema" content="http://schemas.microsoft.com /intellisense/ie5">
    <script language="Javascript">
        function callMethod()
        {
            service.useService("http://localhost/website1/HelloWorld.asmx? wsdl",
"calService"); //创建服务对象
            var parm1 = Form1.all.mul1.value; //获取第一个参数
            var parm2 = Form1.all.mul2.value; //获取第二个参数
            service.calService.callService(callback,"Hello_World",parm1,parm2); //调用方法
        }

        function callback(res)
        {
            if (!res.error)
                Form1.all.returnValue.value=res.value; //判断返回值
            else
                Form1.all.returnValue.value='计算错误'; //计算错误
        }
    </script>
</HEAD>
<body>
    <div id="service" style="BEHAVIOR:url(webService.htc)"></div>
    <form id="Form1" method="post" runat="server">
        <FONT face=宋体><INPUT type=text id=mul1 name=mul1>*<INPUT type=text id=mul2
name=mul2><INPUT style="WIDTH: 50px" onclick=callMethod() type=button value="="><INPUT
type=text id=returnValue name=returnValue></FONT>
```

```
</form>
</body>
</HTML>
```

【运行效果】

本例中 Web 服务的运行效果如图 22-13 所示。JavaScript 调用服务的效果如图 22-14 所示。



图 22-13 本例中 Web 服务的运行效果

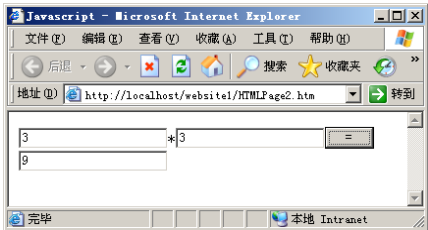


图 22-14 JavaScript 调用服务的效果

【难点剖析】

本例使用了 Web Service.htc 组件，使用此组件的步骤如下：

- (1) 先从微软网站上下载 Web Service.htc。
- (2) 把 Web Service 行为组件绑定到一个 html 标签。
- (3) 用 Web Service 行为组件的 useService 提供 Web Service 的地址。
- (4) 用 Web Service 行为组件的 callService 访问 Webmethod。

注意：本例需要将所有文件添加到 Visual Studio 2005 中运行，因为需要指定 Web Service 的 URL，否则运行出现错误。

22.13 用 JavaScript 实现编码解码

【实例描述】

编码和解码的作用是保护自己的代码或数据不被别人轻易地看到。本例学习如何实现字符的编码和解码。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
```

```

<body>
<script language="javascript">
function tt(obj,str){
    if(str==null){
        obj.value=escape(obj.value);           //编码后的效果
        alert(obj.value);
    }else{
        obj.value=unescape(obj.value);         //解码后的效果
        alert(obj.value);
    }
}
</script>
<textarea name=mytxt cols=30 rows=5></textarea><br>
<input type=button value="编码" onclick=tt(document.all.mytxt)>
<input type=button value="解码" onclick=tt(document.all.mytxt,1)>
</body>
</html>

```

【运行效果】

编码后的文本效果如图 22-15 所示。

【难点剖析】

本例的重点是 JavaScript 提供的两个属性“escape”和“unescape”。两个方法都只包含一个参数，就是要进行编码和解码的字符串。“escape”方法还用于处理页面传递参数中的中文字符。

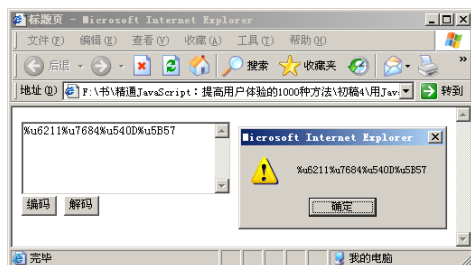


图 22-15 编码后的文本效果

22.14 创建带属性的对象

【实例描述】

JavaScript 一直作为一种客户端语言，早先由于网络流量的瓶颈 JavaScript 面向对象的应用并不广泛，随着 Ajax 和宽带的流行 JavaScript 的面向对象功能也用得越来越多。本例介绍如何创建对象并添加属性。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
    var Person = new Object();           //创建对象
    Person.name = "张三";                //为对象添加属性
    Person.age = "26";
    Person.phone = "010-88888888";
    alert(Person.age );                  //显示对象的属性值
</script>

```



```
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是对象的创建，使用“new Object”完成。对象的属性可以直接用“对象名.属性名”表示，访问这些属性也使用同样的语法。

22.15 用 prototype 实现 JavaScript 的继承

【实例描述】

prototype 是 JavaScript 的框架，可以让 JavaScript 实现面向对象的开发。本例以一个简单的例子，学习如何实现 JavaScript 对象的继承。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
String.prototype.test = function() {           //创建 string 对象的 test 方法
    alert("创建原型对象");
}
var newS = "hello";                             //创建一个新字符串对象
newS.test();                                     //测试新对象是否继承了 test 方法
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点就是 prototype 框架的应用。prototype 被称为原型模式，其真正的定义是“用原型实例指定创建对象的种类，并且通过拷贝这些原型，创建新的对象”。

22.16 JavaScript 制作哈希表

【实例描述】

在保存数据时，使用哈希表可以存储不同数据类型的值。本例通过 JavaScript 创建一个哈希表，学习如何保存数据字典（键/值对）。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```

<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
//自定义哈希表类
function Hashtable()
{
    this._hash = new Object(); // 创建 Object 对象
    //哈希表的添加方法
    this.add = function(key,value){
        if(typeof(key)!="undefined"){
            if(this.contains(key)==false){
                this._hash[key]=typeof(value)=="undefined"?null:value;
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    }
    //哈希表的移除方法
    this.remove = function(key){delete this._hash[key];}
    //哈希表内部键的数量
    this.count = function(){var i=0;for(var k in this._hash){i++;} return i;}
    //通过键值获取哈希表的值
    this.items = function(key){return this._hash[key];}
    //在哈希表中判断某个值是否存在
    this.contains = function(key){ return typeof(this._hash[key])!= "undefined";}
    //清空哈希表内容的方法
    this.clear = function(){for(var k in this._hash){delete this._hash[k];}}
}
var myhash=new Hashtable(); //创建哈希表
myhash.add("name","张三"); //添加键和值
if(myhash.contains("name")) //判断是否存在 name 键
    alert(myhash.items("name")); //根据指定 name 键显示哈希表的值
</script>
</head>
<body>
</body>
</html>

```

【难点剖析】

本例的重点是如何创建哈希表的默认方法，如增加键 值对、移除键 值对、查询键 值等。这些都通过“function”方法实现。要了解哈希表的构造，请参考相关资料。

第 23 章

其他技巧及特效

本章导读

有一些常用的操作技巧，因为篇幅原因没有将其归类。本章逐一列举了实现这些技巧的代码。希望通过这些代码，读者能学习到更多的 JavaScript 相关知识，更深入地了解 Web 的开发技巧。

23.1 最简单的漂移特效

【实例描述】

漂移的文本可以在页面中任意移动。本例是一个完全没有使用特殊效果的漂移，主要学习跑马灯标签 `marquee` 的应用。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<marquee behavior="alternate" height="100%" direction="down">
  <marquee behavior="alternate" direction="right">
    看见这个漂移的效果了吗?
  </marquee>
</marquee>
</body>
</html>
```

【运行效果】

最简单的漂移效果如图 23-1 所示。

【难点剖析】

`Marquee` 元素是 HTML 的标签，可实现其内部文本的滚动特效。其主要的属性“`behavior`”用来控制滚动的样式，主要样式如下所示。

- `behavior=“scroll”`：表示由一端滚动到另一端。
- `behavior=“slide”`：表示由一端快速滑动到另一端，且不会重复。
- `behavior=“alternate”`：默认值——表示在两端之间来回滚动。

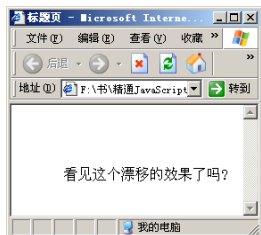


图 23-1 最简单的漂移效果

23.2 JavaScript 遍历对象中的所有属性

【实例描述】

本例学习如何创建对象，如何设置对象的属性，以及如何遍历对象的所有属性。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
  var Person = new Object();           // 创建对象
  Person.name = "张三";                // 为对象添加属性
  Person.age = "26";
  Person.phone = "010-88888888";
```



```
for (var parm in Person)
{
    // 依次显示对象中的所有属性
    alert("属性:  " + parm + " 值:  " + Person[parm]);
}
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是“for...in”循环，此循环通过逐个遍历的方式显示对象的所有属性。访问对象的属性值可以使用索引器的形式，如本例中的“Person[parm]”。通常把“[索引]”的形式称为索引器，括号中的值可以是“0”开始的索引，也可以是唯一的标识值。

23.3 QQ 在线客服

【实例描述】

QQ 在线客服是腾讯集团为网站提供的一种服务，用户可将这些代码添加到网页中，实现网站的在线客服功能。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>标题页</title>
</head>
<body>
    本站客服热线
    <a target=blank href=tencent://message/?uin=453172580&Site=http:www.google.com&Menu= yes >
    <img border="0" SRC=http://wpa.qq.com/pa?p=1:453172580:13 alt="点击这里给我发消息"></a>
</body>
</html>
```

【运行效果】

QQ 在线客服的运行效果如图 23-2 所示。

【难点剖析】

本例的重点是“<a>”元素中的接口，代码是“tencent://message/?uin=453172580&Site=http:www.google.com&Menu= yes”，其中一定要提供 3 个参数，“uin”、“Site”和“Menu”。“uin”参数代表客服的 QQ 号码；“Site”参数则代表本站的 URL 地址；“Menu”



图 23-2 QQ 在线客服的运行效果表示菜单效果。

23.4 查看网站的排名

【实例描述】

要了解自己网站现在的排名情况，可去 alexa 网站查看。本例通过一段简单的代码链接到 alexa 网站，让站长可以轻松地了解自己网站的情况。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<a href="http://www.alexa.com/data/details?amzn_id=&url=www.baidu.com">查看百度的排名</a>
</body>
</html>
```

【运行效果】

网站的排名结果如图 23-3 所示（本例查看的是百度网站）。



图 23-3 网站的排名结果

【难点剖析】

本例的重点是链接到 alexa 网站所需要的参数。“url”参数表示要查看排名的网站，注意此参数不能带“http://”标识。

23.5 定义全局变量

【实例描述】

在高级开发语言（如 C#、Java）中可以很方便地使用“public”等关键字，定义应用程序



中的全局变量，但 JavaScript 的变量只能存在于当前的方法中。本例通过一个简单的方法实现全局变量的定义。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function toGlobal (varName) {
    window.execScript (varName);           // 定义 varName 为全局使用
}
toGlobal('window.varText = "全局变量";'); // 设置全局变量的值
alert(varText);                             // 显示全局变量的值
</script>
</head>
<body>
</body>
</html>
```

【难点剖析】

本例的重点是“window.execScript”方法，execScript 所执行的脚本是针对整个全局域的。将“varText”变量设置为 window 对象的属性，则在全局中都可以调用此变量。

23.6 动态生成金字塔效果

【实例描述】

本例是一个动态生成表格的特效。可将表格输出为金字塔效果。

【实现代码】

```
<html>
<head>
<title>测试</title>
<script language="javascript">
    // 动态创建表格
    var s='<table width="100%" align="center" cellpadding="0" cellspacing="0"> <tr><td
align="center">★</td></tr>','x="★";
    for(var i=0;i<20;i++){
        s+='<tr><td align="center">'+(x=x+"★")+ '</td></tr>' // 循环输出每行
    }
    document.write(s+"<\table>")
</script>
</head>
<body>
</body>
</html>
```

【运行效果】

金字塔效果如图 23-4 所示。

【难点剖析】

本例的重点是使用循环动态创建 20 行，并设置行的“align”属性值为“center”，这样不管页面怎么改变大小，显示的金字塔始终在中间位置。

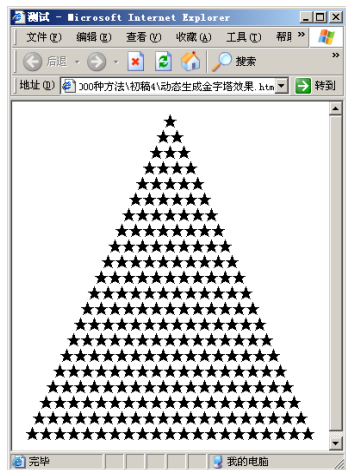


图 23-4 金字塔效果

23.7 动态修改 CSS 的样式

【实例描述】

为了体现用户的操作，一般在用户单击按钮或链接后，需要改变这些控件的样式。这样用户就可以知道已经查看过的相关信息。本例学习如何动态修改控件样式。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function changeStyle(obj)
{
    obj.runtimeStyle.cssText = "color:#990000;border:1px solid #cccccc";
}
</script>
</head>
<body>
<input id="btn1" type="button" value="修改颜色" onclick="changeStyle(this)" />
</body>
</html>
```

【难点剖析】

本例的重点是动态样式的获取。runtimeStyle 对象用来设置控件的格式和样式，并且在运行过程中覆盖已有的格式和样式，其优先级比 style 高，但并不永久保存改变后的样式。

23.8 根据浏览器不同设置 CSS

【实例描述】

不同的浏览器在同样的 CSS 下显示效果可能不同，有时候为了保证界面的一致性，需要根据浏览器的不同设置相应的 CSS。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```



```
<head>
<title>标题页</title>

<script language=javascript>
  if ((navigator.appName == "Microsoft Internet Explorer") && (parseInt(navigator.
appVersion) >= 4))
  {
    document.write('<link rel=stylesheet type="text/css" href="ie.css">')
                                //IE 浏览器下的样式
  }
  else {
    document.write('<link rel=stylesheet type="text/css" href="ns.css">')
                                //Netscape 下的样式
  }
</script>

</head>
<body>
</body>
</html>
```

【难点剖析】

本例并没有给出代码中使用的“ie.css”和“ns.css”样式表。只是说明了如何通过“navigator.appName”属性判断浏览器的类型，然后使用“link rel”动态指定浏览器应该加载的 CSS 样式表。

23.9 汉字按拼音排序

【实例描述】

默认的中文并不是按拼音排序的，本例将学习如何按拼音对指定的中文进行排序。

【实现代码】

```
<html>
<head>
<title>汉字排序</title>
<script language="javascript" >
  function txtSort()
  {
    var txtarray=document.getElementById("txt").value; //获取排序前的文本
    txtarray=txtarray.split(","); //通过间隔符号将文本分隔成数组
    txtarray.sort(); //对数组进行排序
    txtarray.sort(function(txtarray,b){
      return txtarray.localeCompare(b) //使用 localeCompare 进行两个文本的对比
    });
    document.getElementById("txt2").value=txtarray; //显示排序后的结果
```

```
    }
</script>
</head>
<body>
排序前: <textarea id="txt" >占三,里斯,亚当,苹果,香蕉,政治,历史</textarea></p>
<input type="button" id="Button1" value="排序" onclick="txtSort()" /></p>
排序后: <textarea id="txt2" ></textarea></p>
</body>
</html>
```

【运行效果】

排序后与排序前的对比，如图 23-5 所示。

【难点剖析】

本例的重点是“localeCompare”方法。此方法的使用语法如下所示：

```
string.localeCompare(target)
```

其中参数“target”是与“string”进行比较的字符串。如果“string”小于“target”则返回的值小于“0”，如果“string”大于“target”则返回的值大于“0”，如果相等则返回“0”。

23.10 划词搜索

【实例描述】

在一段文本中选择一个单词，然后调用 Google 的搜索功能对此单词进行搜索，这就是本例要实现的划词搜索功能。

【实现代码】

```
<html>
<head>
<title>划词搜索</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<SCRIPT language=Javascript>
document.body.onload=adddiv; //页面加载时动态创建div
document.onmousedown=recordobj; //鼠标按下时触发的事件
document.ondblclick=dbclick; //双击窗体时触发的事件
document.onmouseup=showselect; //鼠标弹起时触发的事件
var eventObj;
var isDouble=false;
var allow=true; //是否启用划词搜索

function isallow()
{
```



图 23-5 排序后与排序前的对比



```

        if(allow){
            allow=false;
            alert('已经关闭');
        }
        else{
            allow=true;
            alert('已经打开');
        }
    }

    function dbclick() //双击事件
    {
        isDouble=true;
    }
    function recordobj() //当前操作对象
    {
        eventObj=event.srcElement;
    }
    function showselect() {
        var str="";
        if(event.srcElement.tagName!="A"&&event.srcElement.tagName!="INPUT" &&event.
srcElement==eventObj&&!isDouble&&allow)
        {
            var oText=document.selection.createRange(); //获取选择文本
            if(oText.text.length>0) //如果文本存在
            {
                str=oText.text;
                oText.text="begin"+oText.text+"end"; //包装被选中的文本
            }
            oText.select(); //实现选择
            //设置选中文本的样式，带下划线，变颜色
            event.srcElement.innerHTML=event.srcElement.innerHTML.replace("begin", "<u
style='FONT-WEIGHT: bold;COLOR: #ff3366'>").replace("end","</u>");
        }
        googleDiv(str) //实现选定文本的搜索
        isDouble=false;
    }

    function googleDiv(str)
    {
        var obj=document.getElementById("googleDiv"); //获取动态添加的 div
        if(str.length>0)
        {
            obj.style.display="block"; //显示 div
            obj.style.position="absolute"; //设置 div 绝对位置
            obj.style.zindex=999;
            obj.style.posTop=document.body.scrollTop+event.y-25; //div 的 y 坐标
            obj.style.posLeft=document.body.scrollLeft+event.x+5; //div 的 x 坐标
        }
    }

```

```
obj.style.widht=80; //div 的宽度
obj.innerHTML="<a target=_blank href=http://www.google.com/search? ie=UTF-8&oe=UTF-8&q="+str+" style='BORDER-RIGHT: royalblue thin solid; BORDER- TOP: royalblue thin solid; FONT-WEIGHT: bold; BORDER-LEFT: royalblue thin solid; CLIP: rect(auto auto auto auto); COLOR: #ffffff; BORDER-BOTTOM: royalblue thin solid; BACKGROUND-COLOR: inactivecaption; TEXT-DECORATION: none'>搜索</a>";
}
else
{
    obj.style.display="none";
}
}

function adddiv()//动态添加 div 标签
{
    var mobj = document.createElement("div"); //创建 div 标签
    mobj.id="googleDiv"; //设置 div 标签的 id
    document.body.appendChild(mobj); //将 div 添加到窗体中
}

</SCRIPT>
<INPUT type="button" onclick="isallow()" value="关闭/打开划词功能">
<p>This is Test ,please Select,it's will google</p>
</body>
</html>
```

【运行效果】

划词后的效果如图 23-6 所示。搜索的效果如图 23-7 所示。

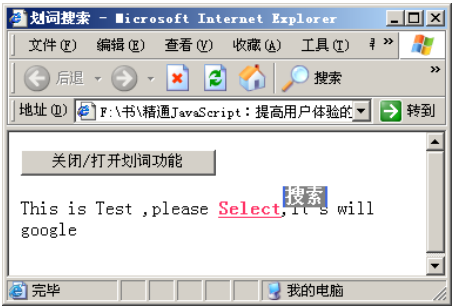


图 23-6 划词后的效果

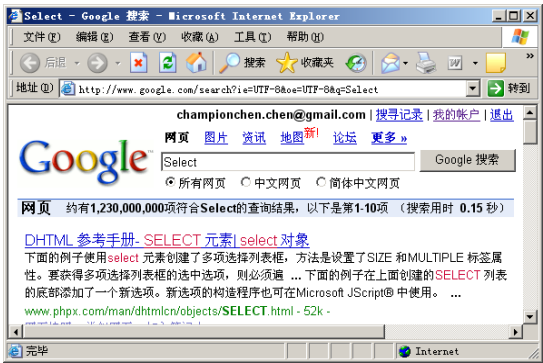


图 23-7 搜索的效果

【难点剖析】

本例的难点有 3 个，获取用户选择的词，动态弹出 div 层让用户调用搜索功能，调用 Google 实现搜索。获取用户选择的词在“showselect”方法中实现，主要依靠“createRange”和“select”两个文本区域选择方法。动态弹出 div 层是常见的层特效，就是设置层的隐藏和显示。调用 Google 搜索的实现在“googleDiv”方法中完成，这里注意传递给 Google 的参数。

23.11 加载大量 input 控件的快速方法

【实例描述】

有些数据编辑页面需要显示大量的 input 控件（200 个以上）。本例学习如何快速显示大量的 input 控件以提高页面的响应速度。

【实现代码】

```
<script language="javascript">
    var d = 200;
    var tmpTableStr=new Array();
    var tempArrayLength=0;
    tmpTableStr[tempArrayLength++]="<table width=\"100%\" border=\"1\" id=aaaa>";
    for(i=1;i<d;i++)
    {
        tmpTableStr[tempArrayLength++]="<tr>";
        tmpTableStr[tempArrayLength++]="<td id=\"td1\" width=40% >";
        tmpTableStr[tempArrayLength++]="<input type=\"text\" id=\"dgip\" name= \"ip1\"
value=\"第一列姓名\">";
        tmpTableStr[tempArrayLength++]="</td>";
        tmpTableStr[tempArrayLength++]="<td id=\"td2\" width=40% >";
        tmpTableStr[tempArrayLength++]="<input type=\"text\" id=\"dgip\" name= \"ip2\"
value=\"第二列年龄\">";
        tmpTableStr[tempArrayLength++]="</td>";
        tmpTableStr[tempArrayLength++]="<td id=\"td3\" width=30% >";
        tmpTableStr[tempArrayLength++]="<input type=\"text\" id=\"dgip\" name= \"ip3\"
value=\"第三列地址\">";
        tmpTableStr[tempArrayLength++]="</td>";
        tmpTableStr[tempArrayLength++]="</tr>";
    }
    tmpTableStr[tempArrayLength++]="</table>"
    document.write(tmpTableStr.join(''));
</script>
```

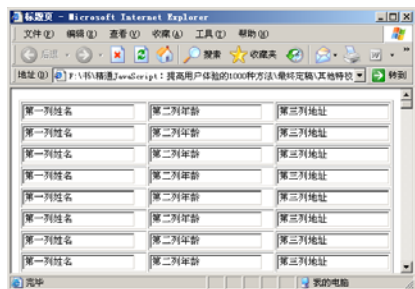


图 23-8 加载 200 个 input 控件后的效果

【运行效果】

加载 200 个 input 控件后的效果如图 23-8 所示。

【难点剖析】

提高速度一直是 Web 应用程序开发的重点，通过本例可以学习两个知识点，如何动态添加 input 和如何提高加载速度。本例中使用 table 控件页面的布局，使用 td 的“width”属性控制每列的宽度，然后使用循环依次添加控件。本例中加载 200 个控件的用时不超过三秒。

23.12 简繁体转换

【实例描述】

字体的变化应用在多语言网站中。本例学习如何实现中文简体和繁体之间的转换。

【实现代码】

```
<script language="javascript">
function window.onload()
{
    //繁体字字库
    var s="省略..... "
    var t="省略..... "
    function String.prototype.stot()//简体到繁体
    {
        var k='';
        for(var i=0;i<this.length;i++)
            k+=(s.indexOf(this.charAt(i))!=-1)?this.charAt(i):t.charAt(s.indexOf(
(this.charAt(i)))
        return k;
    }
    function String.prototype.ttos()//繁体到简体
    {
        var k='';
        for(var i=0;i<this.length;i++)
            k+=(t.indexOf(this.charAt(i))!=-1)?this.charAt(i):s.charAt(t.indexOf(
(this.charAt(i)))
        return k;
    }
}
</script>
```

注意：上述 s 和 t 变量分别是简体和繁体字库，由于书本的印刷问题，本例没有给出，请参考随书光盘。

在 body 中添加两个按钮和一个文本框用来实现转换。

【运行效果】

繁体文本运行效果如图 23-9 所示。

【难点剖析】

本例的重点是使用简体和繁体库，然后使用 String 对象的“indexOf”和“charAt”方法实现逐字节的转换。

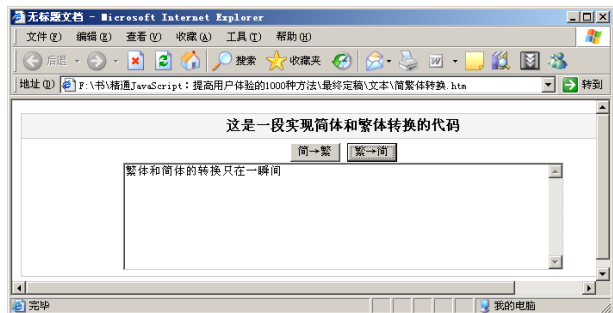


图 23-9 繁体文本运行效果

23.13 将 HTML 转换为 JavaScript 脚本

【实例描述】

随着 C# 的流行，很多时候需要在后台手写 HTML 代码，为了正确地书写，可以使用本例提供的代码自动生成脚本，然后在后台调用。

【实现代码】

```
<script>
    function toScript(val)
    {
        var value = val.value;
        //特殊字符的转换
        value = value.replace(/\\/gi,"\\\\"").replace(/"/gi,"\"").replace(
(/'/gi,"\\'");
        valArr = value.split("\r\n");//分行
        value="";
        for (i=0; i<valArr.length; i++)
        {
            value += (i==0) ? "info =" : " "
            value += " \" + valArr[i];
            value += (i!=valArr.length-1) ? "\" + \"\\n\" + \"\n\" : "\"\\n\" //字符串的间隔符号
        }
        value+="\ndocument.write(info)";
        val.value = value;
    }
</script>
```

需要在 body 中添加一个按钮调用上面的方法，代码如下所示：

```
<input type=button value="将 HTML 转为 JavaScript" onclick=toScript(document. all["mytxt"])>
```

【运行效果】

将 HTML 转换为 JavaScript 脚本的效果如图 23-10 所示。

【难点剖析】

本例的重点是对特殊字符的转换。其中用了“replace”和“split”方法。“replace”用来替换指定的字符，“split”用来将字符串分解为数组，分解的依据就是指定的标识。

23.14 脚本永不出错

【实例描述】

在打开页面时，普通用户并不希望看到弹出脚本错误的对话框，为避免这种情况可以在网页中添加捕获错误的代码。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
  <script language="javascript">
    function killErrors()
    {
      return true;
    }
    window.onerror = killErrors;
  </script>
</head>
<body>
  永远不会出错
</body>
</html>
```

本例没有运行效果，所以并没有提供图例。

【难点剖析】

本例的重点是“window.onerror”。其中 window 是 JavaScript 的常用对象，“onerror”事件是当脚本发生错误时页面触发的事件。

23.15 进入网站的签名程序

【实例描述】

签名程序就是在用户登录网站时要求用户填写能够识别个人身份的登录名。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>标题页</title>
```

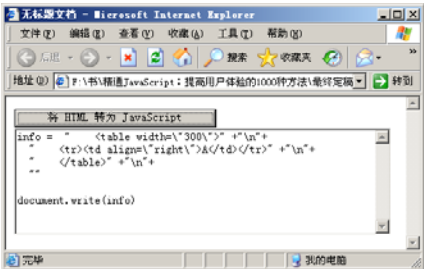


图 23-10 将 HTML 转换为 JavaScript 脚本的效果



```
</head>
<body>
<SCRIPT language="JavaScript">
var name=prompt("您是光临本站的VIP会员,请签名留念,让其他VIP认识你!","初级会员");
document.write("<h3><center>真诚欢迎您来访http://www.google.com<BR><P>" + " " );
document.write("<font color=red>"+name +"</font>"+ " 先生(小姐)</h3>");
document.write(" ")
</SCRIPT>
</body>
</html>
```

【运行效果】

签名的效果如图 23-11 所示。签名后的提示效果如图 23-12 所示。

【难点剖析】

本例的重点是 JavaScript 提供的“prompt”方法。其提供一个文本框,允许用户输入一些信息。本例用“name”变量获取用户输入的信息,然后显示在页面中。

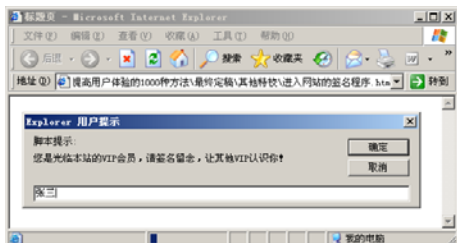


图 23-11 签名的效果



图 23-12 签名后的提示效果

23.16 浏览器毁灭者

【实例描述】

所谓毁灭者是指产生窗口刷屏或不断弹出新窗口的效果。本例学习如何一次弹出多个窗口。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="JavaScript">
function pop(){
    for(i=1;i<=3;i++)    //循环打开三个窗口
    {
        //指定打开窗口的样式
        window.open('http://www.google.com',' ','width=50,height=50','status=off','location=off','toolbar=off','scrollbars=off')
    }
}
```

```

</script>
</head>
<body>
<form name="form">
    <p>
        <input type="button" value="测试:单击就打开 3 个 50*50 的小窗口" onClick= "pop()"
name="button" >
    </p>
</form>
</body>
</html>

```

【运行效果】

弹出窗口后的效果如图 23-13 所示。

【难点剖析】

本例的重点是打开新窗口的方法“window.open”。通过一个“for”语句循环执行“window.open”命令打开 3 个新窗口,且这 3 个窗口的样式都一样。

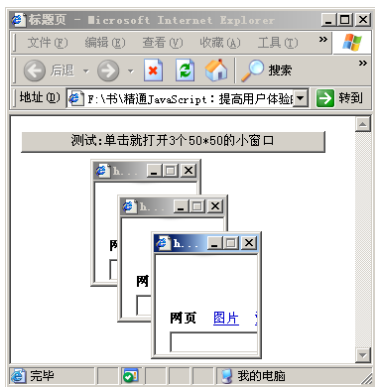


图 23-13 弹出窗口后的效果

23.17 罗列对象的属性和值

【实例描述】

要了解一个对象的使用方法,就得了解对象的属性。本例可单独作为一个 JavaScript 小工具获取对象的所有属性和属性对应的值。

【实现代码】

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language ="javascript">
function GetPros(obj)
{
    for(var pros in obj)                //遍历所有的属性
    {
        alert( pros+"="+obj[pros])    //显示所有的属性值
    }
}
</script>
</head>
<body>
    <input id="Button1" type="button" value="获取属性" onclick="GetPros(this)"/>
</body>
</html>

```

**【难点剖析】**

本例的难点是属性的遍历。在 C#中可以通过“foreach”语句轻松实现遍历，但 JavaScript 只有最基本的“for”语句。本例使用“for”语句遍历按钮控件中的每一个属性，并通过“obj[pros]”索引器方式显示这些属性的值。

23.18 密码保护页

【实例描述】

网站中某些页面只有会员或管理员才可以进入。本例学习如何为网页设置密码保护。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function password()
{
var inputCount = 1;
//弹出密码输入框
var pass1 = prompt('请输入密码(密码默认为 password):','');
//判断输入密码是否超过三次
while (inputCount < 3) {
if (!pass1)
history.go(-1);
if (pass1 == "password") {
alert('密码正确!');
break;
}
inputCount+=1;
//弹出错误信息提示
var pass1 = prompt('密码错误!请重新输入:');
}
if (pass1!="password" & inputCount ==3)
history.go(-1);
return " ";
}
document.write(password());
</SCRIPT>
</head>
<body>
</body>
</html>
```

【运行效果】

要求输入密码的界面如图 23-14 所示。

【难点剖析】

本例的重点是弹出密码输入框。这里不是使用普通页面，而是使用 prompt。prompt 和 alert 一样都弹出一个对话框，不同的是 prompt 要求输入数据，而 alert 则是弹出一些提示信息。

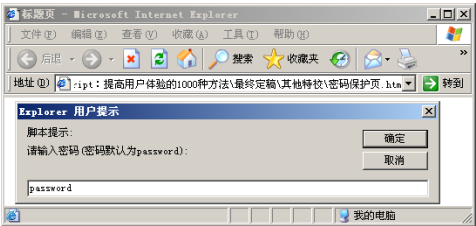


图 23-14 要求输入密码的界面

23.19 全角转半角

【实例描述】

全角是中文输入法下的一种字符形态。为了控件界面的字符效果，本例提供一种全角转半角的方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script LANGUAGE="JavaScript">
function fullChar2halfChar(str)
{
    var result = '';
    for (i=0 ; i<str.length; i++)
    {
        code = str.charCodeAt(i);           //获取当前字符的 unicode 编码
        if (code >= 65281 && code <= 65373)   //unicode 编码范围是所有的英文字母以及各种字符
        {
            result += String.fromCharCode(str.charCodeAt(i) - 65248);
                                           //把全角字符的 unicode 编码转换为对应半角字符的 unicode 码
        }
        else if (code == 12288)               //空格
        {
            result += String.fromCharCode(str.charCodeAt(i) - 12288 + 32);           //半角空格
        }
        else
        {
            result += str.charAt(i);           //原字符返回
        }
    }
    return result;
}
</script>
</head>
<body>
全角<input type="text" name="txt1" value="this is !"><br />
半角<input type="text" name="txt2" value="">
<input type="button" value="转换文本" onClick="txt2.value=fullChar2halfChar(txt1.value)">
</body>
```

</html>



图 23-15 全角转半角后的效果

【运行效果】

全角转半角后的效果如图 23-15 所示。

【难点剖析】

本例的难点是 unicode 编码。可使用字符串对象的“charCodeAt”方法获取某个字符的 unicode 编码，然后根据编码范围逐个检测用户输入的文本，如果编码范围在“65281~65373”之间，则表示当前字符是全角，用此值减去“65248”便是半角符号。

23.20 全屏广告

【实例描述】

在进入一些大型综合网站时，经常会弹出一些广告，有时候是一些全屏广告。本例学习如何制作这种全屏类型的广告。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<script>
var myWin=window.createPopup() //创建一个弹出式窗口
myWin.document.body.oncontextmenu=myWin.document.body.onselectstart=function(){return false}
myWin.document.body.innerHTML="<table bgcolor=black width=100% height=100%><tr><td align=center style=color:white;font-weight:bold;font-size: 75px;cursor:default>这里是全屏广告的内容</td></tr></table>" //输出一个全屏表格
myWin.show(0,0,screen.width,screen.height) //移动到顶端，并全屏
</script></body>
</html>
```



图 23-16 全屏广告的效果

【运行效果】

全屏广告的效果如图 23-16 所示。

【难点剖析】

本例的重点是使用“createPopup”方法弹出一个窗口，然后在新窗口中动态输出一个表格，最后使用“show”方法将新窗口设置为全屏。

23.21 5 秒钟后消失的广告

【实例描述】

在打开大型门户网站时，通常会显示占页面一半的广告，然后等待几秒钟，页面恢复正常。

本例学习如何制作这种定时消失的广告。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
<div id="Ad01">
<iframe src="LOGO.gif" width="430" height="205" frameborder="no" border="0"
marginwidth="0" marginheight="0" scrolling="no">
</iframe></div>
<script language="javascript" type="text/javascript">
function hidetdiv()
{
    document.getElementById('Ad01').style.display="none";    //设置样式为隐藏
}
setTimeout("hidetdiv()",5000);
</script>
<input type="text" name="txt1" value="this is test!">
<input type="button" value="转换文本" onClick="javascript:changeCase(txt1)">
</body>
</html>
```

【运行效果】

带广告的效果如图 23-17 所示。广告消失后的页面效果如图 23-18 所示。



图 23-17 带广告的效果

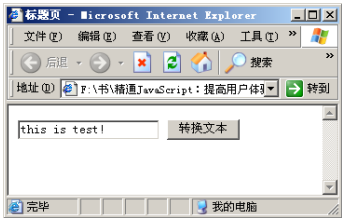


图 23-18 广告消失后的页面效果

【难点剖析】

本例的难点主要包括广告页面的加载和自动消失使用的定时器。广告加载使用的是“iframe”框架，此框架可以使指定页面嵌入到当前页面中。定时器用来每隔一段时间就执行“hidetdiv”方法，此方法的目的是将框架隐藏。



23.22 输入的英文自动全大写

【实例描述】

在有些文档中需要将内容中出现的英文字母都大写，为了方便操作可在网页中将文本框的内容设置为英文自动大写。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
</head>
<body>
输入英文小写字母,测试发生的变化<br />
<input onkeydown="this.value=this.value.toUpperCase()">
</body>
</html>
```

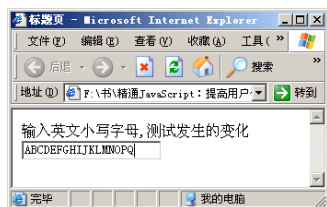


图 23-19 本例的运行效果

【运行效果】

本例的运行效果如图 23-19 所示。

【难点剖析】

本例中首先使用“onkeydown”获取用户输入字母时触发的事件，然后使用字符串对象的“toUpperCase”方法，将当前文本框值“this.value”中的英文全部转换为大写。

23.23 特殊扩散效果

【实例描述】

扩散是经常出现在 Flash 中的效果。本例学习如何使用 JavaScript 实现扩散效果。

【实现代码】

```
<script language="javascript">
var x,y;
var timer;
var i_fontsize=0;
var step=0;
var thisx,thisy;
//根据不同浏览器下的鼠标位置
function handlerMM(e)
{
x = (document.layers) ? e.pageX : event.clientX;
y = (document.layers) ? e.pageY : event.clientY;
}
```

```

function mydivup()
{
if (document.all) {
thisx = x;
thisy = y;
mydivup2();
}
}
//实现特殊效果的方法
function mydivup2()
{
if (i_fontsize<=1000) {
document.all.mydiv.style.fontSize=i_fontsize;
document.all.mydiv.style.color="rgb(255,"+Math.floor(i_fontsize/6)+","+"+Math.floor(i_f
ontsize/6)+")";
document.all.mydiv.style.posLeft=thisx-(Math.floor(i_fontsize/3)); //x 坐标的变化
document.all.mydiv.style.posTop=thisy-(Math.floor(i_fontsize/1.4)); //y 坐标的变化
step+=2;
i_fontsize+=step; //字体逐渐变大
timer=setTimeout("mydivup(2)",50);
}
else {
clearTimeout(timer);
i_fontsize=0;
step=0;
document.all.mydiv.style.posTop=-10000;}
}
document.onmousemove = handlerMM;
</script>
<style>
.mydivstyle {
position:absolute;
visibility:visible;
top:-50px;
font-size:5pt;
font-family:Verdana;
color:FF0000
}
.explain {
position:absolute;
top:80px;
left:40px;
width:300px;
color:000000;
text-align:center;
font-size:20pt;
font-family:Times;
}
}

```

```
</style>
```

需要在 body 中添加一个 div，代码如下所示：

```
<div id="mydiv" class="mydivstyle">
<p><font color="#eec0cc"><<>>></font></p>
</div><div id="redirection" class="explain">
<p><a target="_blank" onMouseOver="mydivup("
href="http://www.google.com">把鼠标移动到此处</a></p>
</div>
```

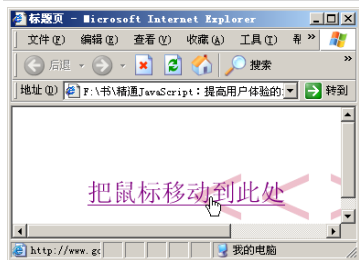


图 23-20 扩散效果

【运行效果】

扩散效果如图 23-20 所示。

【难点剖析】

本例的重点在于扩散效果的实现原理。使用一个定时器随着时间的变化不断增大文本的字体，并同时改变文本的坐标值，从而实现扩散效果。本例中“step”变量用来指示字体变大的步长。

23.24 提交信息等待界面

【实例描述】

网络带宽不足的时候，如果用户与服务器进行交互则会出现白屏或无任何反应等问题，这时可以使用层给予用户提示，让等待状态不再枯燥。

【实现代码】

```
<script language="javascript">
function Save()
{
    msg.style.visibility="visible";//显示
}
</script>
```

需要在 body 中添加一个名为“msg”的 div 层来包装等待信息，还有一个按钮用来调用“Save”方法，代码如下所示：

```
<input id="Button1" type="button" value="保存" onclick="Save()" />
<div id="msg" style="position:absolute; z-index:8; width: 300; visibility: hidden">
<table width="300" height="50" border="0" cellspacing="2" cellpadding="0"
bgcolor="#FFb609">
<tr>
<td bgcolor="#ccffff">数据库正在处理数据，请稍候...</td>
</tr>
</table>
```

</div>

【运行效果】

提交信息等待界面如图 23-21 所示。

【难点剖析】

本例使用了 JavaScript 与 CSS 结合的样式，通过“style.visibility”来显示或隐藏提示信息。当“visibility”属性的值为“visible”时显示信息，为“hidden”时则隐藏信息。

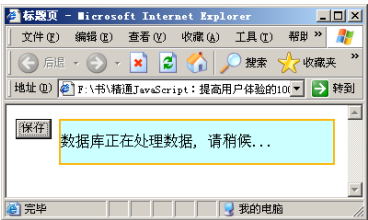


图 23-21 提交信息等待界面

23.25 同时调用两个方法

【实例描述】

在 body 标签中可以使用“onload”事件调用一个方法，如何使此事件同时调用两个方法呢？本例学习如何同时调用两个方法。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function test1()                                //第一个方法
{
    alert("第一个方法");
}
function test2()                                //第二个方法
{
    alert("第二个方法");
}
function together()                             //同时调用的方法
{
    test1();
    test2();
}
</script>
</head>
<body onload="together()">
同时调用两个方法实例
</body>
</html>
```

【难点剖析】

本例的重点是“together”方法。其完成了对两个方法的调用，然后使用 body 标签的“onload”事件调用“together”方法。本例主要以变通的方式实现两个方法的同时调用。

23.26 自定义错误处理样式

【实例描述】

在开发 C/S 应用程序时，可以使用“try...catch”语句来捕获页面的错误，然后使用自定义的错误样式反馈给用户。本例使用 JavaScript 自定义错误样式，这样当浏览器出现错误时，可以显示给用户一些详细信息。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<SCRIPT LANGUAGE="JavaScript">
function errorbox(){
    errorwindow=window.open("", "test", "width=300,height=200");
    errorwindow.document.write('<title>脚本错误报告</title><center>该页面运行过程中发现错误，请通知网站管理人员。<br><form><input type="button" value="关闭窗口" onClick= "window.close()">
</form></center>')
    errorwindow.document.close();
    errorwindow.document.bgColor="#eeb7ff";
    return true;
}
window.onerror=errorbox;                                //调用定义错误的方法
</script>
<script>
alert("该脚本错误，没有右括号")
</script>
</head>
<body>
</body>
</html>
```



图 23-22 错误提示的效果

【运行效果】

错误提示的效果如图 23-22 所示。

【难点剖析】

本例的重点有两个，错误处理的定义和处理事件的绑定。错误处理定义通过“errorbox”完成，其打开一个新窗口并显示提示信息。处理事件的绑定通过“window.onerror”完成，此事件在窗体发生错误时被触发。本例在“btnClick”方法中使用了一个不存在的控件，用来引发错误事件。

23.27 FTP 网站登录

【实例描述】

FTP 是文件传输协议，一般用于往服务器上传送文件。本例简单制作了一个 FTP 网站的登

录界面。

【实现代码】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>标题页</title>
<script language="javascript">
function goFtpSite() { //当前页面的导航, 注意登录地址、用户名和密码
document.location.href = "ftp://" + document.ftp.login.value + ":" + document.
ftp.password.value + "@" + document.ftp.url.value;
}
</script>
</head>
<body>
<form name="ftp">
<table border="0" cellpadding="1" cellspacing="1" bgcolor="#000000" align="center">
<tr>
<td>
<table border="0" cellspacing=0 cellpadding=5 align="center">
<tr bgcolor="#bbbbbb">
<td width="75" align="right">
<font face="arial, helvetica" size="-1">
Ftp://
</font>
</td>
<td>
<font face="arial, helvetica" size="-1">
<input type="text" size=15 name="url">
</font>
</td>
</tr>
<tr bgcolor="#dddddd">
<td align="right">
<font face="arial, helvetica" size="-1">
Login:
</font>
</td>
<td>
<font face="arial, helvetica" size="-1">
<input type="text" size="15" name="登录名" maxlength="20">
</font>
</td>
</tr>
<tr bgcolor="#bbbbbb">
<td align="right">
<font face="arial, helvetica" size="-1">
Password:
</font>
```

```
</td>
<td>
<font face="arial,helvetica" size="-1">
<input type="password" size="15" name="密码" maxlength="20">
</font></td>
</tr>
<tr bgcolor="#ffffff">
<td colspan="2" align="center">
<font face="arial,helvetica" size="-2">
<input type="button" onclick="goFtpSite();" value="登录">
<input type="reset" value="清空">
</font>
</td>
</tr>
</table>
</td>
</tr>
</table>
</form>
</body>
</html>
```

【运行效果】

FTP 的登录效果如图 23-23 所示。登录后的效果如图 23-24 所示。

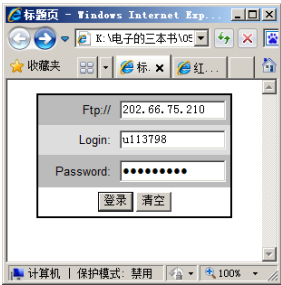


图 23-23 FTP 的登录效果



图 23-24 登录后的效果

【难点剖析】

本例的重点其实是传输协议的设置。如果要定位到普通网站，则“document.location.href”属性要设置为以“http://”开头。如果要定位到 FTP 网站则需要以“ftp://”开头，并且指定登录名和登录密码。